6th INTERNATIONAL CONFERENCE ON INTELLIGENT GAMES AND SIMULATION

GAME-ON 2005

EDITED BY

Marwan Al-Akaidi

and

Abdennour El Rhalibi

NOVEMBER 24-25, 2005

DE MONTFORT UNIVERSITY
LEICESTER, UNITED KINGDOM

A Publication of EUROSIS-ETI

Printed in Ghent, Belgium

Cover art of Divinity 2 was reproduced by kind permission of Larian Studios, Oudenaarde, Belgium

6TH International Conference on Intelligent Games and Simulation

LEICESTER, UNITED KINGDOM
NOVEMBER 24-25, 2005

Organised by
EUROSIS

Co-Sponsored by

Liverpool John Moores University

Binary Illusions

University of Bradford

Delft University of Technology

GIGnews.com

Ghent University

The Moves Institute

Simulation First

Sheffield University

Hosted by

De Montfort University

Leicester, United Kingdom

EUROSIS-ETI Publication

ISBN: 90-77381-23-6

GAME'ON
2005

PREFACE

We extend a hearty welcome to all participants at the *6th GAME-ON 2005 International Conference on Intelligent Games and Simulation*. In its six years of existence, it has rapidly gained recognition worldwide as being one of the first scientific conferences dealing with important research issues surrounding the use of AI and computer simulation in next-generation commercial games. A characteristic, which, is now recognized across the Atlantic, where the first counterpart of the conference was held at McGill University in Montreal, the first of many, we are sure.

This year, modelling and simulating agent behaviour again make up the biggest group of papers, thus enforcing its importance in commercial games research. The topic of gaming environments stimulated intense discussion last year and appears again in this year's programme.

Our Keynote Speaker this year, Jim Parker of the University of Calgary, Calgary, Canada, addresses the different challenges in creating a driving game simulation, which constitutes an area of considerable interest to researchers in computer driving simulation games: As usual, participants will be able to select the best paper for the annual games award from the previously designated papers.

One of the highlights of this year's event is the visit to the new Virtual Reality Suite of the De Montfort University, where a demonstration will be given on the multiple graphical based applications offered by such a versatile system.

We hope participants will have the chance to explore some of the sights and sounds of Leicester and will enjoy this year's games event.

Marwan Al-Akaidi, General Conference Chair
Abdennour El-Rhalibi, General Programme Chair
Leicester, November 2005

CONTENTS

CONTENTS

SCIENTIFIC PROGRAMME

SIMULATION
AND
AI

Carcassonne Java Jess Expert Game
Intelligent Board Games and Query-Based Utility Reasoning

René Molenaar[1], Ludo Maat[1], L.J.M. Rothkrantz[1]

[1] Media and Knowledge Engineering - Faculty of Electrical Engineering, Mathematics and Computer Science
Mekelweg 4
2628 CD Delft
The Netherlands
Renem@ch.tudelft.nl - LudoMaat@zonnet.nl - L.J.M.Rothkrantz@ewi.tudelft.nl

KEYWORDS

Carcassonne, Jess, Java Swing, Artificial Intelligence, Expert Games, Intelligent Query Based Utility Reasoning.

ABSTRACT

Board games like Carcassonne involve lots of rules that players have to obey. In a digital version of such a game a rule based expert system shell like Jess functions as a rule supervisor, while Java manages all the interaction. The game knowledge base can make the board react intelligently and since all the game facts are present in the knowledge base an artificial player can reason and play along. The reasoning is accomplished by adding all the different and sometimes conflicting reasons into a utility score. The artificial player makes a one shot decision on the best move accordingly. Strategy and heuristics can also be implemented as rules. This decision making is guided by a list of weighting factors that allows the AI to be customized and its level scaled. By using a Java command post the program flow is controlled and easily extended into a network game. By adding lots of details like sound and graphical details the game is made more involving.

1. INTRODUCTION

Carcassonne is a well known award winning board game and became game of the year in 2001 in Germany. Carcassonne involves a randomly growing board with different elements. The game like any other game is made up by rules. Making decisions when playing depends on a lot of factors. Sometimes these decisions have conflicting interests. The topic of this paper is a design and implementation of a digital version of the game. Related work can be found on Internet (Wikipedia).
Expert Systems are capable of using rules to assert new facts from a knowledge base. If the game state is in the knowledge base and the rules are in the expert system, the expert system can act as a supervisor. But not only that, the rule engine also makes it possible to make complex decisions by analyzing the knowledge base and running a list of queries thus creating an artificial player.

This is the goal of the project: To use an expert system to make an intelligent computer version of the game Carcassonne. The expert system acts as game supervisor and artificial player(s).

2. BACKGROUND

This project involved bringing three elements together which we will first discuss individually. These elements are:

- The board game Carcassonne,
- Java Swing GUI and
- Jess Expert System.

2.1 Carcassonne

Carcassonne is a turn based board game. The game can be played with two to six players. The game consists of 72 tiles, with each tile having one or more of the following elements: city, road, church and field. The game starts with one tile that shows the elements road, city and 2 fields as seen in figure 1.

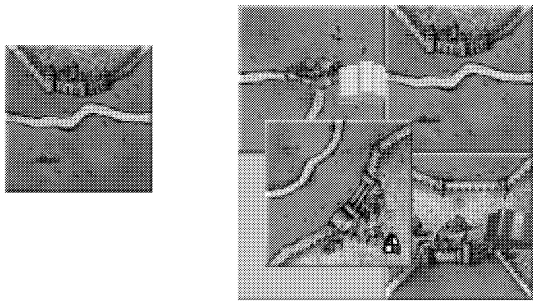


Figure 1. Example Tiles

After the placement of this first tile, the game continues according to the flow in figure 2.

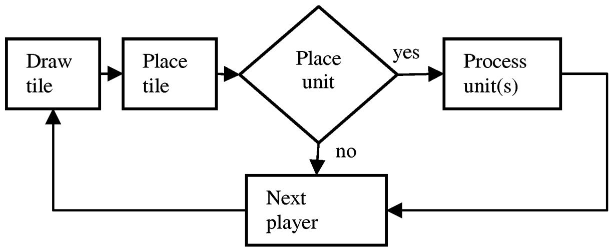


Figure 2. Game Turn Flow

Each turn a player gets one random tile to connect with the placed tiles on the table. This placement has to be valid, that is all roads, fields and cities must be connected between neighboring tiles.

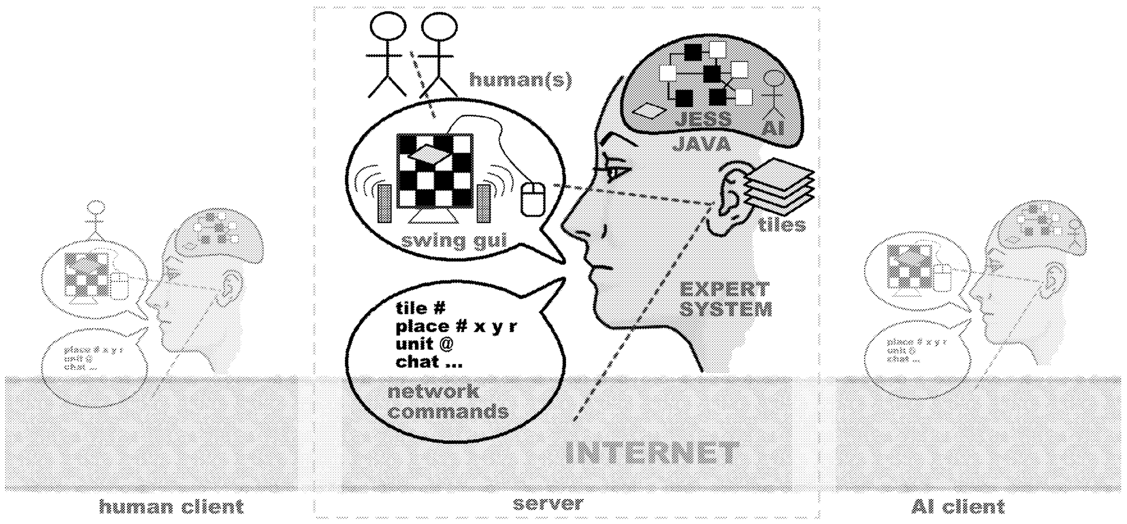


Figure 3. Structural scheme of connected expert systems acting as autonomous entities each with 3 Elements: Jess, Java and Swing GUI. The game logic (i.e. state, rules, and intelligence) is implemented in Jess (the brain). The interactive aspects and network communication are implemented in Java (the mouth and ears). The server draws a random tile from the stack and announces the number, listens till a place command has been issued by the current player or AI, and then announces the issued command. Each networked game instance is responsible for inferring the results of interaction independently

After the tile placement the player gets the opportunity to place a unit (called follower or meeple) on one of the unowned elements of that tile making that player the owner of that element (city, road or field). When an element is completed, the units placed on that element earn points for their owning players and are returned to the players to be stationed again later. Finished elements only produce points for the players with the most units on it. If two or more players tie for the most units, all tied players own that element and score points. Thus the players collaborate in connecting and finishing the elements on the tiles. When there are no more tiles available points are awarded for owning unfinished elements. The units on the fields are counted to produce points for finished cities in the fields. A city can be located in a few unconnected fields. The players contributing the most units added up together scores for a city. The player with the most points wins the game.

2.2 Java Swing for GUI

Java is a very well known cross-platform, reflective, object-oriented programming language from Sun Microsystems. Java provides several toolkits to create Graphical User Interfaces. The two well known toolkits are Java AWT and Java Swing. These toolkits consist of common graphical components used in interfaces, like text fields, message boxes, buttons, scroll bars and mouse listeners. Java Swing includes lightweight graphical components and is one part of the Java Foundation Classes (JFC). It consists of roughly the same functionality as AWT, but the graphical design is fancier. Since Java Swing is written in the Java language, it can run uniformly on every platform providing the same results, unlike Java AWT.

2.3 Jess for Logic

Jess is a rule engine and scripting environment written in Java, created by Ernest Friedman-Hill of Sandia National Labs in 1995. Jess is a descendant of the (rule-based portion of) CLIPS language. CLIPS was developed (by Gavin Riley) starting in 1984 at NASA-Johnson Space Center and provided an expert system extension for the C-language. The powerful scripting language Jess is integrated in the Java environment. Using this Java expert system shell Java software can be extended to reason with declarative rules. It is also possible to create, modify and reason with Java objects without compiling Java code. *Jess is small, light, and one of the fastest rule engines available* (Friedman-Hill 2005). Jess uses an enhanced version of the Rete algorithm to process rules and works with a network of nodes that make up the rules. Rules are activated by facts. But one fact can activate many rules. Activated rules are placed in a conflict set. Rules have to be selected from the conflict set to be fired. This problem can be solved by the Rete Algorithm. Rete has become the basis for many popular expert systems. This algorithm sacrifices memory for increased speed (Treijtel, Waveren van, 2001) an important factor for gaming.

3. CONCEPT AND IMPLEMENTATION

To build the game, two distinct modules are needed. The first module shows the board to the user and handles the interfacing. The second module manages all the game rules and intelligence. See figure 3 for an abstract interpretation of this scheme in which the individual systems are depicted as autonomous persons. The Java Jess combination allowed a smooth integration of the two modules for intelligence and interfacing.

3.1 Java and Jess Integration

The communication between Jess and Java is easily established. Their combination can roughly be done in 5 different ways, varying from Jess with some Java support to Java with some Jess support. Our approach was the middle (completely integrated): Jess manages the intelligent elements;

Java does the rest (i.e. graphics, sounds, mouse etc.). A central command post allows for a controlled communication between the two parts. Few commands are communicated through the command post (see figure 3). These commands are:

- Details on the new drawn tile,
- The location and rotation where a tile is placed and
- If or where a unit is placed.

The command post also allowed upgrading to a network version by simply rerouting the command to remote players.
The command post can be used as a Jess interpreter. A text field and execute button in the user interface allows any Jess statement to be executed directly without recompiling. This way the rules, reasons and state of the board can be manipulated very directly even in the middle of a game.

3.2 Framework

The game flow can be divided in several steps, which are shown in figure 4. During each turn, a player gets a tile from the stack and places it somewhere on the board. The flow of events in each turn is specified in figures 5, for tile and unit phase respectively.

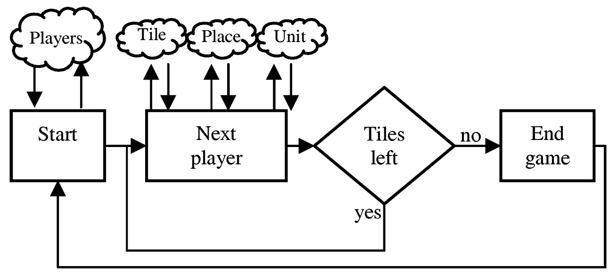


Figure 4. Game Flow

3.3 Adding Intelligence to the Game

Every game is build by rules. When playing games some entity has to check the validity of actions. Jess is used for this purpose. Jess knows the state and the rules of the game and acts as a supervisor, making sure all game rules are applied. Jess prevents illegal moves, indicates options, calculates scores and returns units, and checks connections and ownership of elements. For example Jess has rules to tell Java to highlight potential spots green, invalid spots red and to mark those that are impossible for the rest of the game with an **X**. Jess works with lists that always consists of an enclosing set of parentheses and zero or more symbols, Java objects, numbers, strings, or other lists. For instance a road fact contains a number and a list of tile facts. The (combination of) facts make rules fire. This way some new facts can be asserted. Rules and facts in Jess look like this:

Example facts:
```
(road (nr 2001) (tiles <Fact-170> <Fact-154>))
(roadtofinish <Fact-122> 4 2001)
```

Example rule:
```
(defrule road-finished
  ?road <- (road (nr ?nr)(tiles $?tiles))
  (not (roadtofinish ? ? ?nr))
 =>
  (bind ?points 0)
  (printout t crlf "$$$ road finished: $$$$"  crlf)
  (foreach ?k $?tiles
    (bind ?points (+ ?points 1))
  )
  (assert (finished road ?nr ?points))
)
```

3.4 Adding Intelligent Reasoning Opponents - Query Based Utility Reasoning Score

We will now spend some time on the rules needed to place units and tiles by giving an example. Our game has many – some more complicated than other - rules for the placing of tiles and units. Generally the following happens in two phases:

```
First a place for the tile is sought
  for each found place:
    run all specific utility-queries
    make a checkutil by adding the scores * factors
    if it is better then previously found spots then
        bind this spot to ?result.
        check if a unit should be placed.
  place tile on ?result.

Then the unitphase is handled:
  if placeunit or good grasspot
      place unit
  else
      no-unit
```

The reasoning is done by running a list of queries (rules without an antecedent) and adding their score times factor into a positive or negative utility score. The queries represent reasons similar to those of a human expert player. Sometimes some queries don't have to be executed, for example when a player has no units. And sometimes some specific scores are made 0 again, for example when sneaking a unit in someone's field the take-new-score is 0 again, because the unit will be used otherwise. The factors are mainly used by the programmer to focus AI on certain aspects of the game. This is for testing and debugging new reasoning and for indicating the importance of tactics (like cooperating or taking risks) according to the expert programmer. The factor and score products (i.e. the utility scores of specific reasons) are manipulated by (player) specific properties: how many units are left, how many tiles are left in the game. Some utility scores are limited to a maximum or minimum. For example the risk utility makes sure that difficult places are avoided, this score is limited, else helping an enemy might have a less negative score. In order for all utility queries to be compact a few extra rules and functions are implemented. This is needed to extract facts from the current situation. For example: checking what can be connected, counting units, checking who will win and finding out what new places can be started. The Jess code that causes reasoning looks like this:

Example factor:
```
(defglobal ?*helpmine* = 20)
```

Example query:
```
(defquery extendmine
"search extend fact with current player ruler"
 (declare (variables ?G ?PNUM ))
 ?G <- (SEARCH::found ?place ?ii ?rt)
 (extend ?place ?x1 ?ii ?rt ?score1 ?nr1 TRUE
 $?counts1)
 (test (total-build-ruler ?PNUM $?counts1))
)
```

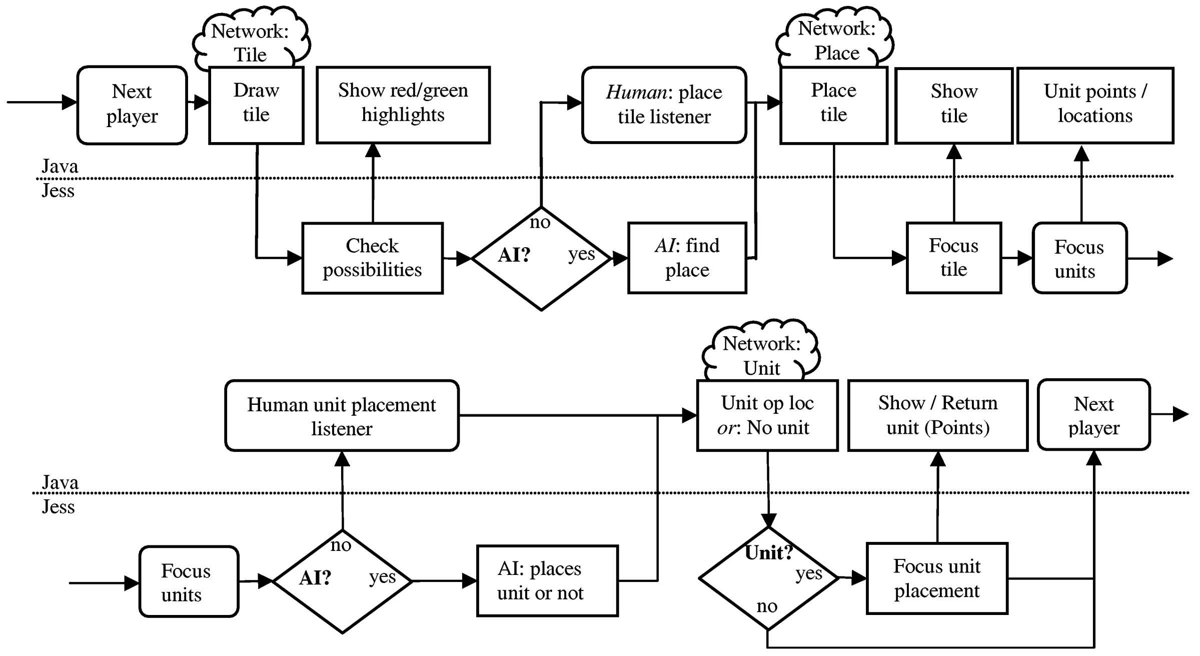


Figure 5 - Place Tile and Unit Phase - Java shows the Graphical User Interface, Jess checks possibilities, updates the new interconnections in the fact database after tile placement and calculates the unit scores and their return. Jess also controls the AI reasoning

Example reasoning:

```
(bind ?ul (get ?*player* unitsleft))
(bind ?utilmine (* (- 7 ?ul) ?*helpmine* (count-
 query-results extendmine ?factid ?pnum)))
```

Each computer player turn the search-best-place-rule runs all queries and manipulates all scores to find the best spot and rotation. Strategy and heuristics can also be implemented as rules. When there are only a few tiles left, the artificial player will focus more attention on owning the fields if there are enough units available.

3.5 Adding Network Capabilities

One of the goals of this project was to provide the game with network capabilities to play across LAN or Internet. Every remote player uses its own Jess Engine and keeps a local copy of the board and the tiles to be drawn in memory. Using this separation it is only needed to broadcast the three types of events that were discussed in section 3.1 to all remote players.

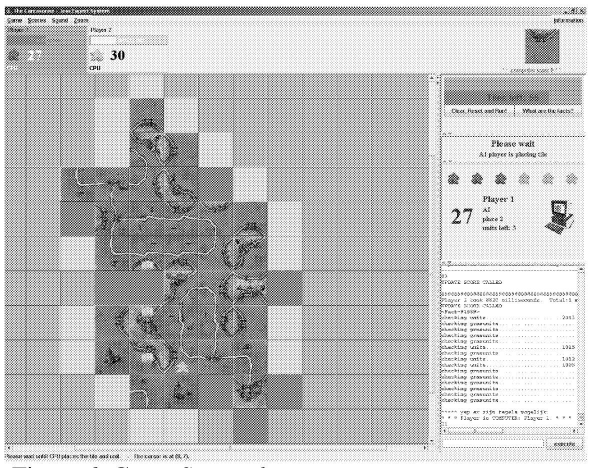


Figure 6. Game Screenshot

4. RESULTS

We implemented a Carcassonne Expert System (see figure 6) that acts as game supervisor and artificial player(s). We tested, we played and we improved by trial and error. The rules of the intelligent board are easily manipulated and extended even in the middle of a game. Intelligent opponents are a challenge and can also be manipulated and extended in a simple manner. The challenge depends on the players' skill and the number of players, but generally the skill of the artificial opponent is better then we had expected. Human players tend to take bigger risks and can be very lucky sometimes or very unlucky or they make one fatal mistake. The AI players tend to have more steady results (the content of the 2 player high-score tables is shown in figure 7). It takes an expert player to be able to defeat the artificial players time and again. The Artificial players are much faster in making their decisions then most humans. We added sounds, icons, messages, game manual, zooming functions, high score system and network capabilities and this resulted in a game that can be executed cross platform.

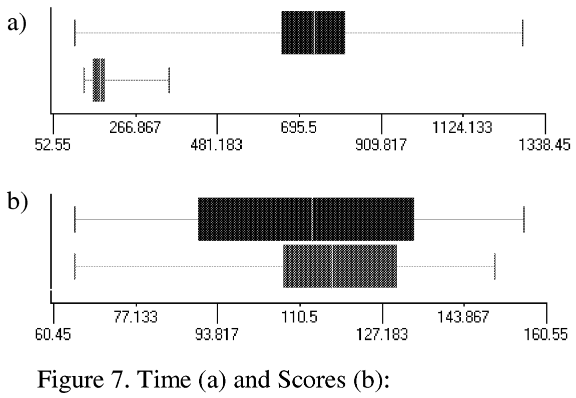


Figure 7. Time (a) and Scores (b):
a) Time needed in Seconds to complete the game for a human (top) and AI (bottom) player and
b) Final Scores for human (top) and AI (bottom)

5. CONCLUSION

We have described the framework and concept of making a digital intelligent version of the turn-based board game Carcassonne with intelligent opponents and shown how the different elements of reasoning and interfacing are combined. The board reacts intelligently on interaction and acts as a rule supervisor. The independently operating rules made the creation of the digital board game straightforward by translating all the game rules into Jess statements. The existence of the game logic in separate facts made the addition of artificial players possible. The implementation of game logic was far less complicated then adding the feature of AI players. Reasoning is not a straightforward translation of rules and takes expert game knowledge.

Difficulties with AI
It was complicated to test and debug our system since changes in the reasoning files could have unknown effects on the average score. It is possible that the players had luck so multiple tries had to be run to draw conclusions. Debugging the reasoning is difficult, because if a strange move occurs, it takes some time to locate the flaw. The errors are mostly quite small. Facilities to test the same complex situation time and again and better output on decisions are therefore a welcome feature for the future.

The reasoning system is not foolproof. More rules could improve the difficulty level of the artificial players, but will also make the AI more complex and harder to analyze. Most reasoning is based on simple risk analyses and the programmer's expertise of the game. In this particular game it seems to be more important not to make (big) mistakes then to make the best possible decision.

Drawbacks of Reasoning System
The main drawback of this first prototype is that all decisions are made in a one-shot fashion. The utility-score does not doubt twice, it settles with the highest utility score. Human players might think twice before committing. The computer most of the times 'forgets' what he was doing and bases the decision on the current situation, the only thing that is 'remembered' is sneaking into grass. And it is **here** that our **weak point** is **located:** Having a *versatile future plan* that involves more than one or two tiles is beyond the capabilities of our current reasoning system. This might sometimes play a role in a 2 player game, because you will get many choices and tiles, with more players most *good* decisions are based on the current situation. On the other hand, human players tend to help weak players and sabotage strong player. The AI does not see any difference (yet). It tries to find the best place at that moment for itself. This plays an important role in the final score with multiple opponents. In the near future adding strategy and heuristics similar to (Treijtel 2001) (Waveren van, 2001) will have high priority. This can be realized in the rule based way of Jess. Rules can for example be placed in a category with common dynamic salience.

The scoring system doesn't translate easily to good reasons; sometimes the least negative score is the reason. Sometimes scores might differ from -10000 to +10000, but usually scores are in the -500 + 500 range. And finally compared to playing with humans, the AI won't feel the pleasurable satisfaction of winning or care about loosing, though speech samples offer a wonderful illusion of these human feats.

Odds versus shrewdness
The computer is far better at calculating the chances and the scores; AI for example doesn't forget that there are no units left in the players' possession (a common mistake made by human players, especially in a computer version of the game). However being wicked can drastically change the odds. The original game-manual mentions that other players may give opinions and suggestions about placements, a factor that is lost: mentioning one place over another is not possible against AI. Instead the human player is helped (a lot) by coloring potential places green, which also makes the game go faster. It helps a lot to know the capabilities of the AI, it is possible in some cases to fool or trick the AI on its weak points. It does however always make an interesting and involving match, the AI has one goal: to win the match.

Advantages
The combination of Java and Jess is very robust and versatile. It is very easy to add and delete rules to the engine. Each rule stands alone and can be manipulated separately. These things can be done with a build in console even in the middle of a game (i.e. without compiling). The reasoning of AI-players is also easily adaptable, because it is a series of loose factors that are added together, new factor can be inserted alongside. Factor can be manipulated independently to create different levels of challenge, changing this dynamically after judging skills or providing these options to players is one of the future goals. Currently the goal considering artificial players was to play the best possible game.

Making use of a Java command post allowed the game to be extended into a network game very simply. We learned that 'the little things' like sounds and graphical details are a very important factor in making a game fun and involving.

Left for the Future:
There are some possibilities left for the future like undo option, difficulty levels / manipulating factors, environment reasoning agents instead of a one-shot score system, more dynamical strategies, saving complex situations and better output on reasoning.

Jess 7 will have a lot of improvements that could be exploited to improve the implemented game: better run-query facilities, slot-specific statement, for each statement and many more new features that would have been very convenient.

6. REFERENCES

Friedman-Hill, E. - Jess – The Rule Engine for the Java platform official website: `http://herzberg.ca.sandia.gov/jess/`

Molenaar, R.C.; Maat, L.; Gangadajal, R.-"Carcassonne: Making an intelligent agent to play the game Carcassonne", EWI Report 07/2005

Riley, G - Clips – A tool for building expert systems Official website: `http://www.ghg.net/clips/CLIPS.html`

Treijtel, C.; L.J.M. Rothkrantz "Stratego Expert System Shell", Proceedings of the 2001 GAME ON, 17-21

Waveren van, J.M.P.; L.J.M. Rothkrantz "Artificial Player for Quake III Arena", Proceedings of the 2001 GAME ON, 48-55

Wrede K.J. - Carcassonne –Klaus Jürgen Wrede website: `http://www.carcassonne.de`

Wikipedia, the free encyclopedia Official website: `http://en.wikipedia.org/`

Hybrid fuzzy system and Fuzzy behaviour implemented in Computer Go

P.Lekhavat & C.J. Hinde
Computer Science, Research School of Informatics
Loughborough University, United Kingdom
P.Lekhavat@lboro.ac.uk, C.J.Hinde@lboro.ac.uk

Abstract
In the game of Go, especially in an opening stage, most expert players rely on their sense and reasoning, which is commonly represented in linguistic terms. Instead of calculating the stone potential in a hard crisp value, they usually state their estimation as this group being light or heavy and use that as a reason for their actions. This paper explains an approach using fuzzy reasoning to simulate a human players knowledge, which the system would be able to acquire and tune. In addition, fuzzy behaviour can be learned and would simulate characteristics which would be useful in a field of computer games.

Key words: Computer Go, Fuzzy, Fuzzy reasoning, Fuzzy behaviour, Hybrid system, AI, Machine learning.

1. Introduction

Go is a strategy board game [BGA, 1999], well known in far east, also known as Wei Qi (Chinese), Igo (Japanese) and Baduk (Korean). The game is normally played on a 19x19-grid board. Two players take turns to place a stone on the board. The objective of the game is to secure the most area on the board to win the game.

2. Computer and a game of Go

Computer Go has been developed for more than 20 years. However, it has not achieved the level of an amateur human player yet. In the 19x19 game, an average number of the reasonable games is about 10^{200}, from an average of 10 plausible choices per move for an average 200 moves per game. In the opening stages there are close to 400 possible moves and so even a short look ahead is prohibitively large. There needs to be some mechanism for selecting a small number of plausible moves so Artificial Intelligence or Machine Learning is needed. The main objective is to develop the system that is able to learn and play Go at high human player level. To achieve that goal, the main approach is using a hybrid AI system to imitate the way that human players think and learn.

2.1 Territory and Potential

In the game of go, the winner is determined by the player with higher territory at the end of the game. However, in early stage of the game, because most groups of stones are not totally settled and there is far too much open space, positional judgment needs to rely on both territory and potential of the group. Territory is an area inside the group that likely to be surrounded. Whereas, potential is the outside area that is influenced by the stones.

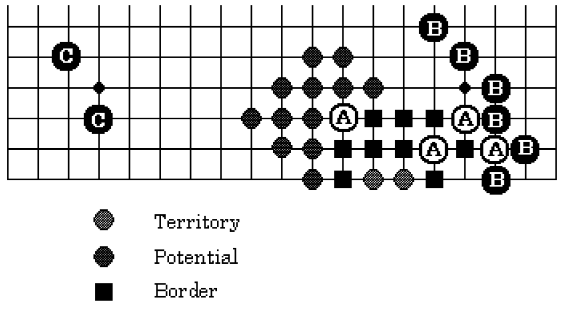


Figure 1 Territory and Potential of stones group A

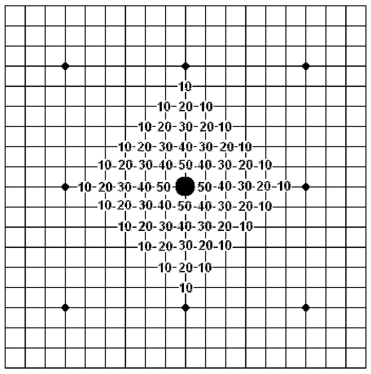
2.2 Fuzzy Influence

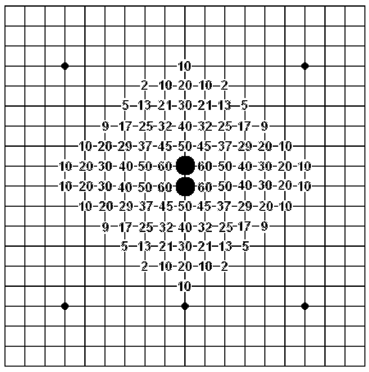
An area inside a group of stones is considered as its prospective territory. On the other hand, the outside area can be considered as the potential of the group where it can expand in the future. As the stones are placed, other empty spaces receive influence from the stones near by. Since the influence effected from the stones, a space near a larger string receives higher influence. The following algorithm is used to calculate stone influence. Compare to Zobrist's [Zobrist, 1970] and Ryder's [Ryder, 1971], an advantage of this algorithm is that the stones potential is extended when more stones are placed together.

Step 1: All liberties around the stones are given the value of 30

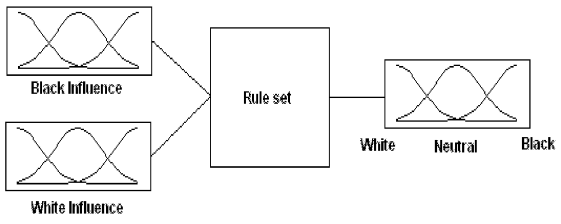
Step 2: Regarding to stones near by, increase a liberty value by 20 for each stone in contact position, and 10 for stone in diagonal position.

Step 3: Value of extended liberties are decreased by 10 until less than 0 or blocked by another stone. In addition, if an extended liberty is shared by 2 sources, the value is average of the source values.





With the results from influence algorithm, Black and White influence values are compared, according to rule set, to determine whether that coordinate is under Black, White or neutral potential, unlike most algorithms that use sum of positive and negative value.



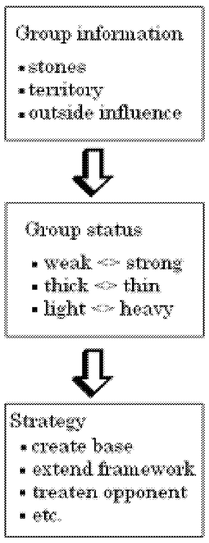
3. Fuzzy reasoning and Hybrid system

Unlike conventional computing and searching methods as often used in chess, the human Go players usually describe stones or groups of stones with linguistic words in their reasoning; such as weak, strong, light or heavy, by setting local objectives for each group of stones then compare them as a global view to find the best operation for the situation.

3.1 Fuzzy reasoning: determined group status and strategy

Commonly, game status is judged by the stones position. Positional judgment usually relies on basic information; stones, territory and influence [Yen, 2001]. If they were accurately estimated, it is possible to use a fuzzy system to justify each group status then summarise the local objective or strategy for each group.

According to the size of the group, its territory and its potential, the group can be described using Go terms such as weak, strong, thick, thin, light or heavy. Each type of the group has its own characteristic which can be used as a guide for the player's objective; carry on by the operations or actions that can be done with or to the group.



3.2 Hybrid fuzzy system

Although fuzzy reasoning capable of deducing the objective, it is only one part of the cycle. Sub-objectives suggested by Fuzzy reasoning module need to be matched up with a pattern move that support the objective. The pattern recognition module explores pattern moves stored in database regarding to objectives that suggested by the fuzzy module. As the result, search trees can be created and used to find the best move or the best variation.

As for the fuzzy rulebase, it should be able to improve. Starting from the initial state, rulebase can be developed by learning from the games that have been played or games from professional human players.

3.3 Fuzzy behaviour

Even though the best objective can be very clear in some situations, most situations have no clear answer. Even the top professional players can have a very different opinion of the same situation, which they can spend hours arguing with no result, especially in early stage of the game. That because each player always has there own preferences and style, one may prefer territory before potential or vice versa. On the other hand, the game usually continues for one or two hundred moves, so the result of the game cannot be used as a measurement since there usually several mistakes occur during the game. In addition, from the view of computer game, if the computer player is always using the same style and playing a predictable move, the human player can feel bored, does not matter how good the computer player is.

Character-based AI for computer games has been discussed before as in [Isla, 2002]. With the learning ability for the hybrid fuzzy reasoning system, it is possible to generate different style of play by adjusting rulebase with the learning system. The idea is to learn the game from a certain player; the rulebase can reflect the style of the player that it uses as a model.

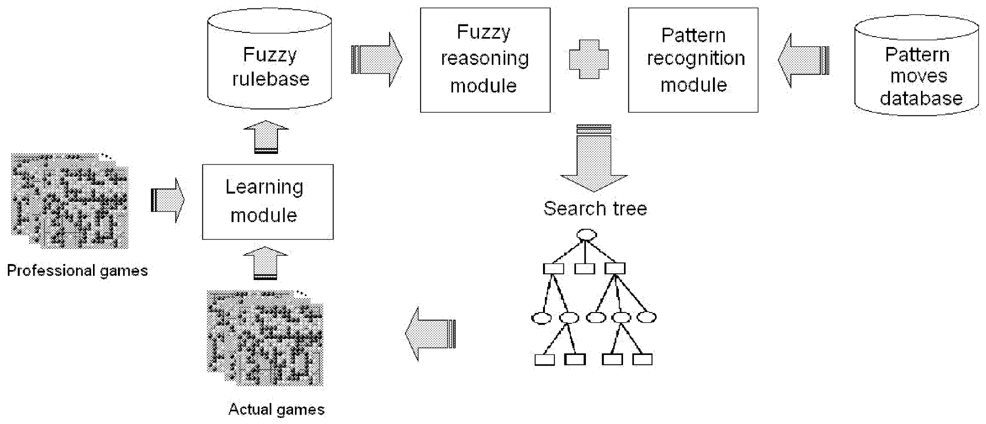


Figure 2 Hybrid system cycle

3.4 Experiment and future work

So far, the project is in an early stage, apart from perception level, about half of the fuzzy system module has been done. However, it is capable of determining and selecting operations for groups of stones, concerning only local area environment regarding to initial set of rulebase. Although the test was limited, the results were very acceptable. As for future work, global reasoning and pattern recognition are most important parts which are expected to be done. However, this system concentrates on early stage of the game, so-called opening game. The system needs an extended module to assist the search tree before it can play a complete game.

4. Conclusion

This paper is concerning with a way to develop computer games. Although it is implemented to a certain board game called Go, it can be adapted to other kinds of games that need AI as well as other fields of use such as expert systems. The ideas suggested in this paper are

- New way of calculating stone influence for game of Go
- How fuzzy reasoning can be used in Game of Go
- A model of Hybrid fuzzy system that capable of reasoning, searching and learning
- Fuzzy behaviour that can be useful in developing of AI computer game

References

[BGA, 1999] British Go Association. Go: The most challenging board game in the world. London, United Kingdom, 1999.

[Bozolich, 2001] R. Bozolich. The Go players almanac. Kiseido publishing company. Tokyo, Japan 2001.

[Bozolich, 2002] R. Bozolich. Five hundred and one opening problems. Kiseido publishing company. Tokyo, Japan 2002.

[Zadeh, 1965] L.A.Zadeh. Fuzzy sets. Journal of Information and Control, 8:338-353, 1965.

[Zobrist, 1970] A.Zobrist. Feature extractions and representation for pattern recognition and the game of Go. PhD thesis, Graduate School of the University of Wisconsin. Wisconsin, United States, 1970.

[Ryder, 1971] J. Ryder. Heuristic analysis of large trees as generated in the game of Go. PhD thesis, Department of Computer Science, Standford University, 1971.

[Burmeister, 2000] J.M.Burmeister. Study in human and computer Go: Assessing the game of Go as a research domain for cognitive science. PhD thesis, School of Computer Science and Electronical Engineering and School of Phychology, The University of Queensland, Queensland, Australia 2000.

[Yen, 2001] S.J. Yen, S.C. Hsu. A positional judgment system for computer Go. Advance in computer games, 9:313-326, 2001.

[Isla, 2002] D. Isla, B. Blumberg. New Challenges for Character-Based AI for Games, MIT Media lab, Cambridge, 2002.

Optimising Reinforcement learning for neural networks

Evan Hurwitz
School of Electrical and Information Engineering
University of the Witwatersrand
Johannesburg, Gauteng, South Africa
e.hurwitz@ee.wits.ac.za

Tshilidzi Marwala
School of Electrical and Information Engineering
University of the Witwatersrand
Johannesburg, Gauteng, South Africa
t.marwala@ee.wits.ac.za

Abstract – *Reinforcement learning traditionally utilises binary encoders and/or linear function approximators to accomplish its Artificial Intelligence goals. The use of nonlinear function approximators such as neural networks is often shunned, due to excessive difficulties in implementation, usually resulting from stability issues. In this paper the implementation of reinforcement learning for training a neural network is examined, being applied to the problem of learning to play Tic Tac Toe. Methods of ensuring stability are examined, and differing training methodologies are compared in order to optimise the reinforcement learning of the system. TD(λ) methods are compared with database methods, as well as a hybridised system that combines the two, which outperforms all of the homogenous systems.*

Keywords: reinforcement, learning, temporal, difference, neural, network, tic-tac-toe.

1 Introduction

Artificial intelligence (A.I.) can only truly be considered worthy of the name when the system in question is capable of learning on its own [1], without having an expert *teacher* available to point out correct behaviour. This leads directly into the paradigm of *reinforcement learning* [1]. The advantage of using such techniques for gaming A.I. is that it would allow a gaming agent to actually learn while in-game, and adapt its own play to that of the player. Most *reinforcement learning* techniques explored utilise binary system representations, or linear function approximators, which severely hinder the scope of learning available to the artificial intelligence system. One notable exception is the work by G. Tessauro on *TD-Gammon*, in which he successfully applied the TD(λ) reinforcement learning algorithm to train a neural network, with staggeringly successful results. Following attempts to emulate his work have, however, been met with failure due to the extreme difficulties of combining backpropagation with TD(λ). These difficulties, and some solutions to them, are explored in this paper, with the A.I. system being applied to learn to play the game of tic-tac-toe by playing against itself. The reason for Tic-Tac-Toe as a choice of games is deliberately because of its simplicity, as the commonly occurring problems become easier to identify within the simpler system, and solutions are then also easier to develop.

2 Background

It is necessary to understand the workings and advantages of neural networks to appreciate the task of applying them in the reinforcement learning paradigm. It is likewise important to fully grasp the implications of reinforcement learning, and the break they represent from the more traditional supervised learning paradigm.

2.1 Neural network architecture

The fundamental building-blocks of neural networks are *neurons* [2]. These neurons are simply a multiple-input, single-output mathematical function [2]. Each neuron has a number of *weights* connecting inputs from another layer to itself, which are then added together, possibly with a *bias*, the result of which is then passed into the neuron's *activation function* [2]. The activation function is a function that represents the way in which the neural network "thinks". Different activation functions lend themselves to different problem types, ranging from yes-or-no decisions to linear and nonlinear mathematical relationships. Each *layer* of a neural network is comprised of a finite number of neurons. A network may consist of any number of layers, and each layer may contain any number of neurons [2]. When a neural network is run, each neuron in each consecutive layer sums its inputs and multiplies each input by its respective weight, and then treats the weighted sum as an input to its activation function. The output will then be passed on as an input to the next layer, and so on until the final output layer is reached. Hence the input data is passed through a network of neurons in order to arrive at an output. Figure 1 illustrates an interconnected network, with 2 input neurons, three hidden layer neurons, and two output neurons. The hidden layer and output layer neurons can all have any of the possible activation functions. This type of neural network is referred to as a *multi-layer perceptron* [2], and while not the only configuration of neural network, it is the most widely used configuration for regression-type problems [3].

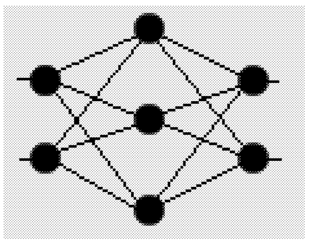


Figure 1. Sample Connectionist network

2.2 Neural network properties

Neural networks have various properties that can be utilised and exploited to aid in the solving of numerous problems. Some of the properties that are relevant to this particular problem are detailed below.

2.2.1 Universal approximators

Multi-layer feedforward neural networks have been proven to be universal approximators [2]. By this one refers to the fact that a feedforward neural network with nonlinear activation functions of appropriate size can approximate any function of interest to an arbitrary degree of accuracy [2]. This is contingent upon sufficient training data and training being supplied.

2.2.2 Neural networks can generalise

By approximating a nonlinear function from its inputs, the neural network can learn to approximate a function [2]. In doing so, it can also infer the correct output from inputs that it has never seen, by inferring the answer from similar inputs that It has seen. This property is known as *generalisation* [2]. As long as the inputs received are within the ranges of the training inputs, this property will hold [2].

2.2.3 Neural networks recognise patterns

Neural networks are often required to match large input/output sets to each other, and these sets are often noisy or even incomplete [2]. In order to achieve this matching the network learns to recognise patterns in the data sets rather than fixate on the answers themselves [2]. This enables a network to 'see' through the data points and respond to the underlying pattern instead. This is an extended benefit of the generalisation property.

2.3 Reinforcement learning

Reinforcement learning involves the training of an artificial intelligence system by trial-and-error, reflecting the same manner of learning that living beings exhibit [4]. Reinforcement learning is very well suited to episodic tasks [4], and as such is highly appropriate in the field of game-playing, where many episodes are encountered before a final result is reached, and an A.I. system is required to evaluate the value of each possible move long before a final result is achieved. This methodology allows for online learning, and also eliminates the need for expert knowledge [4].

2.3.1 Rewards and Returns

Any artificial intelligence system requires some sort of goal or target to strive towards [2], [4]. In the case of reinforcement learning, there are two such quantities that need to be defined, namely *rewards* and *returns* [4]. A reward is defined to be the numerical value attributed to an individual state, while a return is the final cumulative rewards that are returned at the end of the sequence [4]. The return need not necessarily be simply summed, although this is the most common method [4]. An example of this process can be seen below in Figure 2, where an arbitrary Markov process is illustrated with rewards given at each step, and a final return at the end. This specific example is that of a *Random walk* problem.

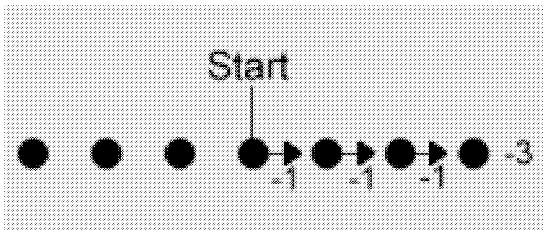


Figure 2. Random walk rewards and returns.

An A.I. system utilising reinforcement learning must learn to predict its expected return at each stage, hence enabling it to make a decision that has a lower initial reward than other options, but maximising its future return. This can be likened to making a sacrifice in chess, where the initial loss of material is accepted for the future gains it brings.

2.3.2 Exploitation vs Exploration

An A.I. system learning by reinforcement learning learns only through its own experiences [4]. In order to maximise its rewards, and hence its final return, the system needs to experiment with decisions not yet tried, even though it may perceive them to be inferior to tried-and-tested decisions [4]. This attempting of new approaches is termed *exploration*, while the utilising of gained knowledge to maximise returns is termed *exploitation* [4]. A constant dilemma that must be traded off in reinforcement learning is that of the choice between exploration and exploitation. One simple approach is the *ε-greedy* approach, where the system is *greedy*, ie attempts to exploit, in every situation with probability ε, and hence will explore with probability 1- ε [4].

2.3.3 TD (λ)

One common method of training a reinforcement learning system is to use the TD (λ) (Temporal Difference) algorithm to update one's value estimates [4] [5]. This algorithm is specifically designed for use with episodic tasks, being an adaptation of the common Widdrow-Hoff learning rule [5]. In this algorithm, the parameters or *weights w* to be altered are updated by equation (1) [5].

$$\Delta w = \alpha\left(P_{t+1} - P_t\right)\sum_{k=1}^{t} \lambda^{t-k} \nabla w P_k \qquad (1)$$

The prediction P_{t+1} is used as a target for finding the error in prediction P_t, allowing the update rule to be computed incrementally, rather than waiting for the end of the sequence before any learning can take place [5]. α and λ are the learning rate and weight-decay parameters, respectively.

2.4 Neural network advantages

Neural networks have a number of advantages that can be exploited in order to optimise a reinforcement learning A.I. system. Some of these advantages are:

- Faster learning. Due to the generalisation property of neural networks, learning from one position can be generalised to learn for all similar positions.
- Positional understanding. As a result of the pattern recognition property of neural networks, the system can learn to judge positions, rather than simply remember individual state configurations.

For these reasons it is desirable to utilise a neural network as a function approximator for a reinforcement learning system.

3 Implementation

In this section we detail the implementation of the problem domain and the various methods of optimising the neural network reinforcement learning, as well as the performance measures used to evaluate the usefulness of each presented method.

3.1 Tic Tac Toe

The game of *Tic Tac Toe*, or *noughts and crosses*, is played on a 3x3 grid, with players taking alternate turns to fill an empty spot [6]. The first player places a 'O', and the second player places a 'X' whenever it is that respective player's turn [6]. If a player manages to get three of his mark in a row, he wins the game [6]. If all 9 squares are filled without a winner, the game is a draw [6]. The game was simulated in Matlab, with a simple matrix representation of the board.

3.2 Player evaluation

The A.I. system, or *player*, needs to be evaluated in order to compare different players, who have each learned using a different method of learning. In order to evaluate a player's performance, ten different positions are set up, each with well-defined correct moves. Using this test-bed, each player can be scored out of ten, giving a measure of the level of play each player has achieved. Also important is the speed of convergence – ie how fast does each respective player reach its own maximum level of play.

3.3 TD(λ) for backpropogation

In order to train a neural network, equation (1) needs to be adapted for use with the backpropogation algorithm [7]. The adaptation, without derivation, is as follows [7]:

$$w_{ij}^{t+1} = w_{ij}^{t} + \alpha \sum\nolimits_{K \in O} \left(P_K^{t+1} - P_K^{t}\right) e_{ijk}^{t} \qquad (2)$$

where the eligabilities are:

$$e_{ijk}^{t+1} = \lambda e_{ijk}^{t} + \delta_{kj}^{t+1} y_i^{t+1} \qquad (3)$$

and δ is calculated by recursive backpropogation

$$\delta_{ki}^{t} = \frac{\partial P_k^t}{\partial S_i^t} \qquad (4)$$

as such, the TD(λ) algorithm can be implemented to update the weights of a neural network [7].

3.4 Stability issues

The TD(λ) algorithm has proven stability for linear functions [6]. A multi-layer neural network, however, is non-linear [2], and the TD(λ) algorithm can become unstable in some instances [4] [8]. The instability can arise in both the actual weights and in the predictions [4] [8]. In order to prevent instability, a number of steps can be taken, the end result of which is in most cases to limit the degree of variation in the outputs, so as to keep the error signal small to avoid instability.

3.4.1 Input/Output representation

The inputs to, and outputs from, a typical A.I. system are usually represented as real or integer values. This is not optimal for TD(λ) learning, as the values have too much variation. Far safer is to keep the representations in binary form, accepting the dimensionality trade-off as a fair price to ensure a far higher degree of stability. Specifically in the case of the outputs, this ensures that the output error of the system can never be more than 1 for any single output, thus keeping the mean error to within marginally stable bounds.

For the problem of the Tic Tac Toe game, the input to the network is an 18-bit binary string, with the first 9 bits representing a possible placed 'o' in each square, and the second 9 bits representing a '1' in each square. The output of the network is a 3-bit string, representing a 'o' win, a draw and an 'x' win respectively.

3.4.2 Activation Functions

As shown in section 2, there are many possible activation function that can be used for the neural network. While it is tempting to utilise activation functions that have a large scope in order to maximise the versatility of the network, it proves far safer to use an activation function that is limited to an upper bound of 1, and a lower bound of zero. A commonly used activation function of this sort is the sigmoidal activation function, having the form of:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

This function is nonlinear, allowing for the freedom of approximation required of a neural network, and limiting the upper and lower bounds as recommended above. While this activation function is commonly used as a middle-layer activation function, it is unusual as an output layer activation function. In this manner, instability is further discouraged.

3.4.3 Learning rate

As the size of the error has a direct effect on the stability of the learning system, parameters that directly effect the error signal also have an effect on the said stability. For this reason, the size of the learning rate α needs to be kept low, with experimental results showing that values between 0.1 and 0.3 prove safe, while higher values tend to become unstable, and lower values simply impart too little real learning to be of any value.

3.4.4 Hybrid stability measures

In order to compare relative stability, the percentage chance of becoming unstable has been empirically noted, based upon experimental results. Regardless of each individual technique presented, it is the combination of these techniques that allows for better stability guarantees. While no individual method presented gives greater than a 60% stability guarantee (that is, 60% chance to be stable given a 100-game training run), the combination of all of the above measures results in a much better 98% probability of being stable, with minor tweaking of the learning rate parameter solving the event of instability occurring at unusual instances.

4 The players

All of the players are trained using an ε-greedy policy, with the value of ε = 0.1. i.e. for each possible position the player has a 10% chance of selecting a random move, while having a 90% chance of selecting whichever move it deems to be the best move. This selection is done by determining all of the legal moves available, and then finding the positions that would result from each possible move. These positions are sequentially presented to the player, who then rates each resultant position, in order to find the best resultant position. It obviously follows that whichever move leads to the most favoured position is the apparently best move, and the choice of the player for a greedy policy. The training of each player is accomplished via *self-play*, wherein the player evaluates and chooses moves for both sides, learning from its own experiences as it discovers errors on its own. This learning is continued until no discernable improvement occurs.

As a benchmark, randomly initialised networks were able to correctly solve between 1 and 2 of the posed problems, beyond which one can say genuine learning has indeed taken place, and is not simply random chance.

4.1 Player #1 – Simple TD(λ)

Player #1 learned to play the game using a simple TD(λ) backpropogation learning algorithm. This proved to be very fast, allowing for many thousands of games to be played out in a very short period of time. The level of play achieved using this method was however not particularly inspiring, achieving play capable of solving no more than 5 of the 10 problems posed in the rating system. The problem that is encountered by player #1 is that the learning done after each final input, the input with the game result, gets undone by the learning of the intermediate steps of the next game. While in concept the learning should be swifter due to utilising the knowledge gained, the system ends up working at cross-purposes against itself, since it struggles to build its initial knowledge base, due to the generalisation of the neural network which is not present in more traditional reinforcement learning arrangements.

4.2 Player #2 – Historical database learning

In this instance, the player learns by recording each position and its corresponding target, and storing the pair in a database. Duplicate input data sets and their corresponding targets are removed, based on the principle that more recent data is more accurate, since more learning has been done when making the more recent predictions. This database is then used to train the network in the traditional supervised learning manner. A problem encountered early on with this method is that early predictions have zero knowledge base, and are therefore usually incorrect. The retaining of this information in the database therefore taints the training data, and is thus undesirable. A solution to this problem is to limit the size of the database, replacing old data with new data once the size limit is reached, thus keeping the database recent. This methodology trains slower than that employed to train player #1, making long training runs less feasible than for TD(λ) learning. The play level of this method is the lowest of those examined, able to solve only four of the ten proposed problems. Nonetheless, the approach does show promise for generating an initial knowledge base from which to work with more advanced methods.

4.3 Player #3 – Fact/Opinion DB learning

Building on the promise of Player #2, a more sophisticated database approach can be taken. If one takes into account the manner of the training set generation, one notes that most of the targets in the database are no more than *opinions* – targets generated by estimates of the next step, as seen in equation (1) – while relatively few data points are in fact *facts* – targets generated by viewing the end result of the game. In order to avoid this problem, the database can be split into two sub-databases, with one holding facts, and the other holding opinions. Varying the sizes, and the relative sizes, of these two sub-databases can then allow the engineer to decide how much credence the system should give to fact versus opinion. This method proved far more successful than Player #2, successfully completing 6 of the 10 problems posed by the rating system. It's speed of convergence is comparable to that of Player #2.

4.4 Player #4 – Widdrow-Hoff based DB learning

In this instance, a very similar approach to that of Player #2 is taken, with one important distinction: Instead of estimating a target at each move, the game is played out to completion with a static player. After each game finishes, the player then adds all of the positions encountered into its database, with the final result being the target of each position. This means that no *opinions* can ever enter into the training, which trades off speed of convergence for supposedly higher accuracy. This method is not optimal, as it loses one of the primary advantages of reinforcement learning, namely that of being able to incorporate current learning into its own learning, hence speeding up the learning process. Unsurprisingly, this method trains with the same speed as the other database methods, but takes far longer to converge. It achieves a similar level of play as does Player #1, being able to solve 5 of the posed problems.

4.5 Player #5 - Hybrid Fact/Opinion DB TD(λ) learning

The logical extension to the previous players is to hybridise the most successful players in order to compensate for the failings of each. Player #5 thus utilises the Fact/Opinion database learning in order to build an initial knowledge base from which to learn, and then proceeds to learn from thence using the TD(λ) approach of Player #1. This proves more successful, since the intrinsic flaw in player #1's methodology lies in its inability to efficiently create a knowledge base, and the database method of player #3 creates that knowledge base from which to learn. Player #5 begins its learning with the expected sluggishness of database methods, but then learns much faster once it begins to learn using the TD(λ) approach. Player #5 managed to successfully solve seven of the ten problems once trained to convergence. The problem of unlearning learned information is still apparent in Player #5, but is largely mitigated by the generation of the initial knowledge base.

4.6 Player comparisons

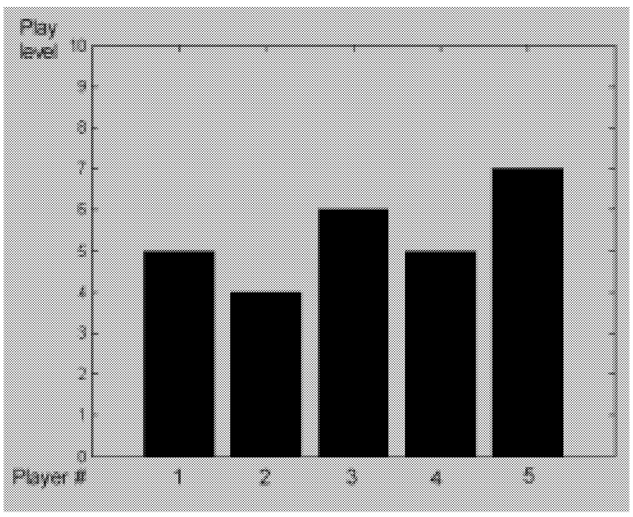


Figure 3. Relative player strengths

As is illustrated in figure #3, the hybrid method learns to play at the strongest level of all of the methods presented. Due to the drastic differences in speed and computational power requirements, it is preferable to stay away from database-based methods, and it is thus worth noting that only the fact/opinion database method arrives at a stronger level of play than the simple TD(λ)-trained player #1, and that this methodology can easily be incorporated into a TD(λ) learning system, which produces the far more promising player #5. The fact that after a short knowledge-base generation sequence the hybrid system uses the highly efficient TD(λ) approach makes it a faster and more reliable learning system than the other methods presented. As can be seen in Figure 3, however, there is still a greater level of play strength that should be achievable in this simple game, and that has been limited by the unlearning error seen in Players #1 and #5.

For future work, it is recommended that research goes into a more efficient method of calculating eligability traces, since a better calculation of the eligibility traces would solve the problem of unlearning encountered in the TD(λ) methods, removing the play-level limit that has been encountered.

5 Conclusion

While the stability of TD(λ) methods for neural networks is in question, many precautions have been presented that greatly improve the stability of the neural network learning process. Far more important is the generalisation property of neural networks. The generalisation that is the great boon of neural networks also is its greatest weakness when applied to reinforcement learning. The ability to generalise comes at the cost of updating all position prediction with each individual prediction, which limits the development of a knowledge base. Database building techniques are useful, but also lose some of the benefits of reinforcement learning, and are hence undesirable. Lastly, the usage of database techniques to develop an initial database, followed by further learning utilising the TD(λ) approach leads to the best player in terms of player strength, illustrating the need for a teacher to 'show the ropes' to an A.I. system, before it is capable of learning on its own.

For future work, the issue on *unlearning* needs to be further explored and solved before neural network reinforcement learning is in fact a viable tool for use in gaming A.I., although once this is accomplished, the scope of the method holds great promise to allow truly adaptable A.I. agents in games, able to give players a true challenge.

References

[1] Sutton, R. S. *What's wrong with Artificial intelligence*, Last viewed 24/08/2005 http://www.cs.ualberta.ca/~sutton/IncIdeas/WrongWithAI.html 11/12/2001.

[2] Wesley Hines J. *Matlab supplement to Fuzzy and Neural Approaches in Engineering.* John Wiley & Sons Inc. New York. 1997.

[3] Miller, Sutton and Werbos. *Neural Networks for Control.* MIT Press. Cambridge, Massecheussettes. 1990

[4] Sutton, R. S, Barto A. G. *Reinforcement learning: An Introduction*, MIT press, 1998.

[5] Sutton, R. S. *Learning to predict by the methods of temporal differences.* Mach. Learning 3, (1988), 9-44.

[6] Morehead A. H, Mott-Smith G, Morehead P. D. *Hoyle's Rules of Games, Third revised and updated edition*. Signet book. 2001.

[7] Sutton, R. S. *Implementation details of the TD(λ) procedure for the case of vector predictions and Backpropogation*. GTE laboratories technical note TN87-509.1, 1989.

[8] Sutton, R S. *Frequenlty asked questions about reinforcement learning*, last viewed 24/08/2005, http://www.cs.ualberta.ca/~sutton/RL-FAQ.html

SIMPLE GAMES THAT TEACH ARTIFICIAL INTELLIGENCE

Alasdair Macleod
University of the Highlands and Islands
Lews Castle College
Stornoway
Isle of Lewis
UK
HS2 0BW
Email: Alasdair.Macleod@lews.uhi.ac.uk

KEYWORDS

Games, Artificial Intelligence, Education

ABSTRACT

It is proposed that by a fundamental exploration of AI techniques using interesting but simple games and puzzles, games development students will become more aware of the potential advantages and the limitations of each of the established methods in common use. The puzzle of Sudoku and the game of Pegboard Solitaire are explored in this context. The expectation is that the application of AI to computer games may consequently become realistic, with the result that effective intelligent systems may be realised. The adoption of this approach for a group of students is reported and the findings may be of interest to educators in general and games technology course designers in particular.

INTRODUCTION

There is currently a strong perception that Artificial Intelligence (AI) should be incorporated into modern computer games to enhance interest. However the impact of such innovation has been limited and it is interesting to consider the reasons [1]. Whilst our relatively poor understanding of the nature of intelligence and the disappointing progress up to now in developing methods that simulate intelligence is certainly a major reason for this, we propose in the paper that another factor is a failure by some practitioners to appreciate the basic limitations of the AI techniques that are in common use. Thus AI methods are often applied with greater expectation than is reasonable.

This conclusion is apparent from the observation of game design and programming students developing their basic skills and knowledge in a programming environment. It is frequently the case that the student will choose an AI technique without fully understanding how it works or what it can do in the context of the application. The technique is nevertheless applied (often after diligent study) to a complex computer game in the expectation of enhanced performance. There is a failure to recognise that no established AI technique is generic and all are application dependent.

In recently reorganising the BSc degree programme at the University of the Highlands and Islands (www.uhi.ac.uk), it was decided to try and address this issue by presenting the AI teaching modules entirely in the context of games development. The subject had previously been taught in the context of commerce, engineering, computing etc. with little reference to games.

The basic underlying principle is that the application of specific AI methods to complex computer games should only occur at the end of a long process where the technique has been fully characterised through extensive application in more controlled, simpler games environments where quantitative testing is possible. This may be through the use of simple highly-constrained games. Although it is possible to directly create AI modules for online games [2], this is not generally useful to the learner as the programming is complex, and it is often extremely difficult to evaluate the effectiveness of AI in such a complex environment. It is preferable to select simple well-constrained games where the effectiveness of an AI implementation is clearly measurable. For the interest of the students, it is desirable to select topical subject matter and annually modify the material to reflect what is 'fashionable'. To illustrate this point, in this paper we will evaluate both the traditional game of pegboard solitaire and the number puzzle Sudoku as suitable AI teaching environments rather than using clichéd examples such as tic-tac-toe [3].

We also comment on the use of games as a general tool to teach computing. Many students who choose to study computing are interested in computer games and indeed this may be the original inspiration for their career choice. Perhaps for this reason (amongst others), it has been found that the incorporation of the study, design and implementation of computer games into the curriculum (or used a vehicle for presenting theory) results in accelerated learning. Of particular interest is the GameMaker program created by Mark Overmars [4] that is exceptionally good for teaching an understanding of game structure and design in the context of games development, but also illustrates object-orientated principles.

SUDOKU

Sudoku is a number placement puzzle that has become something of a craze over the last year [5]. The problem in its normal form consists of a 9 x 9 rectangular grid of cells where each cell must be occupied with a single digit between 1 and 9. In the completed array every column and row must consist of a permutation of the available digits. In addition, the 9 x 9 grid is divided into nine 3 x 3 regions, and each region must contain one and only one occurrence of the digits from 1 to 9. The problem is initially presented with numbers already in some of the cells, the quantity given ranging from a total of 20 for very hard puzzles to 35 for the easy puzzles. The puzzle is solved by placing the correct digits in the empty cells by the application of logic and the use of deduction.

There is a great deal of literature on the puzzle and how AI may be used to both solve and create puzzles; however the focus in this paper is on the value of the game as an educational tool to teach the basic principles of AI. This is particularly appropriate to students who wish to apply their AI knowledge to games in general and Internet games in particular. We will show that the puzzle is a good vehicle for exploring the foundational principles. One should also mention that before AI methods can be applied, a user interface must be developed using some programming language. The game environment is almost as simple as tic-tac-toe but much more interesting. The development of an environment is a useful exercise for teaching elementary programming - the problem is easily represented and demands the application of object orientated principles and a good understanding of multidimensional arrays.

Mathematically, the puzzle is related to scheduling and timetabling, Latin squares, magic squares, path-finding problems, and map-colouring problems. In formal terms, Sudoku is a constraint problem and there has already been significant research conducted using established constraint programming methods [6, 7]. In most cases, this AI technique is more advanced that that encountered in starting undergraduate AI computing courses and should be introduced at a later stage. We recommend that students should not at this stage be expected to discover the best technique, but instead be asked to implement a variety of very simple methods and critically evaluate these.

One of the most important and fundamental methods used in AI is search and the first approach suggested to students after a user interface has been created is to try solve the problem following a methodical search procedure. The purpose of the exercise is to determine if a search procedure is feasible and if so, whether a breadth-first, a depth-first or if some pruning method is appropriate. In fact, even the crudest depth-first search can be used to solve the problem in less than second on a home PC with an algorithm implemented in C++. This is because the constraints grow as numbers are added thus the number of possibilities down the chain decreases rapidly. Student may be encouraged to join Sudoku programming forums, where the programming methods are discussed and interesting (and effective) search techniques are presented. Whilst apparently adequate, pure number-crunching techniques are too slow when trying to create puzzles (as we will see further on), particularly as all tree branches must always be executed to test for duplicate solutions. It is therefore necessary to explore other approaches even though the student may consider their search process adequate.

Because the puzzle of Sudoku represents a formal task, and a unique solution can be found by adopting formal reasoning, the next objective for the student might be to identify the rules human beings apply to solve the puzzle and represent these in a formal way. A restricted search over the game space can then be conducted by applying the derived rules. This will lead to a rapid solution. The method can be related to a very simple Expert System, and used to illustrate the operation of the inference engine. However, in reality, the rules are easily implemented through procedural programming, though it is more illuminating to implement the solution using an AI language such as LISP or Prolog.

Unfortunately the puzzle is fundamentally flawed in that it can be solved by a simple inverse elimination method. If all the cells are temporarily filled with each number from 1 to 9, and the impossibilities eliminated by constraints imposed as cells are occupied with given numbers, the puzzle becomes much easier. A program to implement this procedure is easily written. A student variant is shown in Fig. 1. Many programs of this type are available off the Internet, although the competent student programmer should be able to replicate the core functionality of these in a few hours. Once the given numbers are entered, the program output shows the allowed numbers for the remaining cells (Fig. 1(a)). In many cases, singletons (only possibilities) become immediately apparent. As these numbers are entered (Fig. 1(b)) more singletons appear and the puzzle is often solved with no reasoning or thought (Fig. 19c)).

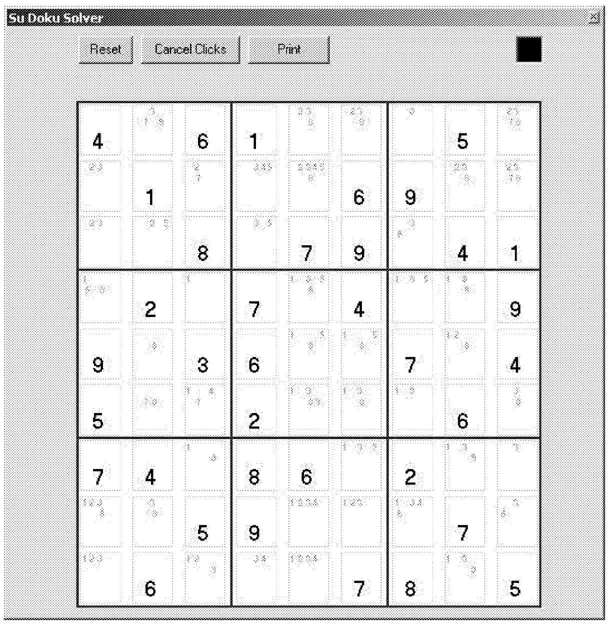
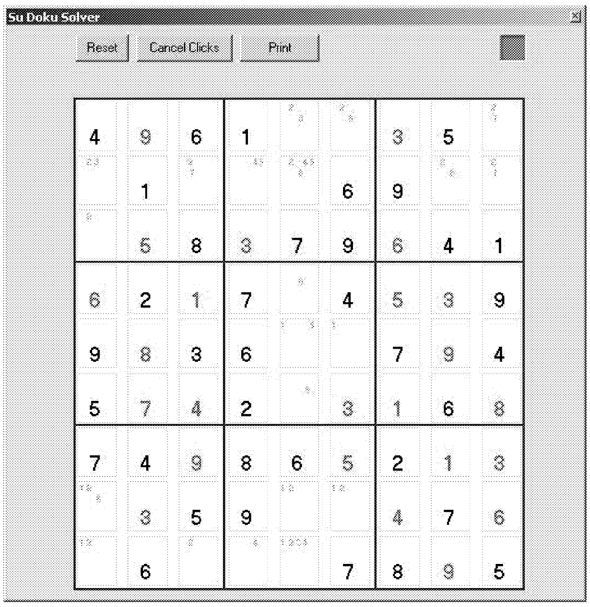
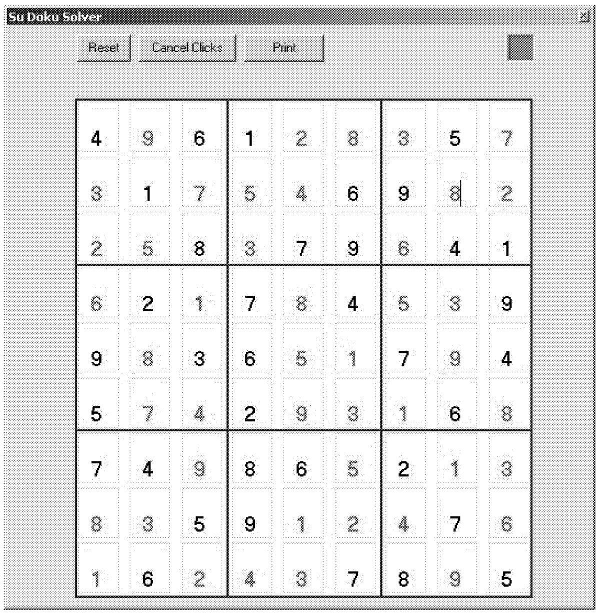


Figure 1 (a): Initial puzzle with the given digits already imposing restrictions on the empty cells.



4	9	6	1			3	5	
	1				6	9		
	5	8	3	7	9	6	4	1
6	2	1	7		4	5	3	9
9	8	3	6			7	9	4
5	7	4	2		3	1	6	8
7	4	9	8	6	5	2	1	3
	3	5	9			4	7	6
	6				7	8	9	5

Figure 1 (b): After the cells with one and only one option selected



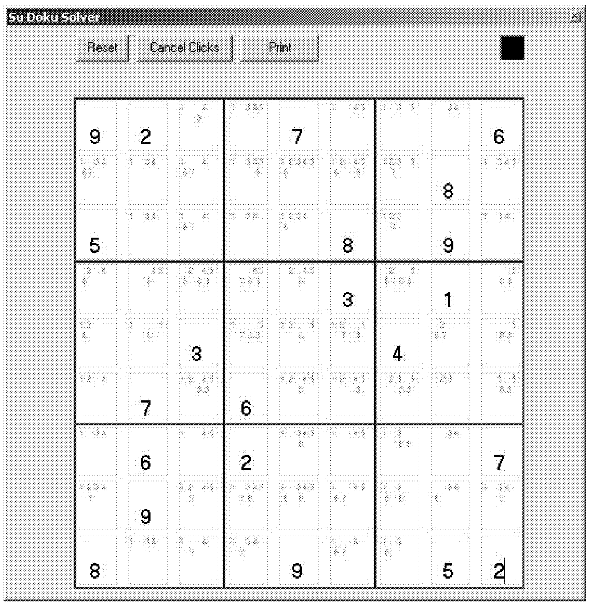
4	9	6	1	2	8	3	5	7
3	1	7	5	4	6	9	8	2
2	5	8	3	7	9	6	4	1
6	2	1	7	8	4	5	3	9
9	8	3	6	5	1	7	9	4
5	7	4	2	9	3	1	6	8
7	4	9	8	6	5	2	1	3
8	3	5	9	1	2	4	7	6
1	6	2	4	3	7	8	9	5

Figure 1 (c): Final Solution

Whilst the method removes the elements of memory and simple reasoning in basic to moderate puzzles, it is not entirely automatic and appears to fail with difficult Sudoku puzzles such as the one illustrated in Fig. 2.

When the supplied digits are entered, it is apparent there are no cells that can now be filled by direct observation. However, the problem is always solved through the application of two simple rules:

1. Look for a number in a candidate number in a row, column or group that occurs just once.
2. Identify contingencies, narrowing a group of numbers to the same number of cells, thus they cannot be in other cells in the set of which the restricted group is a subset.



9	2			7				6
							8	
5					8		9	
					3		1	
		3				4		
	7		6					
	6		2					7
	9							
8				9			5	2

Figure 2: Example of a difficult Sudoku puzzle.

Rule 1 is shown in Fig. 3(a) and rule 2 is illustrated in Fig. 3(b).

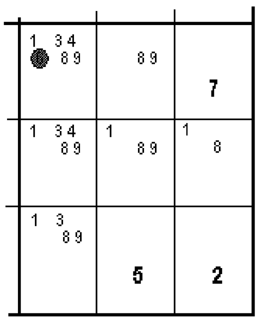


Figure 3(a) Look for instances of a number appearing once is a column, row or group. In this case 6 in the marked cell.

Interestingly, the contingency technique is closely related to the Karnaugh-Veitch method of obtaining simplified Boolean expressions from logic truth tables [8]. The student would be expected to implement these rules to generate an extremely fast solver.

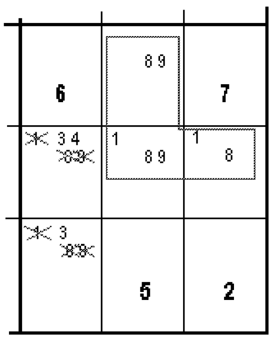


Figure 3(b) An example of contingencies. 1, 8, and 9 must be permuted in the red-bordered squares and these cannot be valid possibilities in the other empty vertical cells.

Much of the attraction of the puzzle is lost when it is seen to be solvable in such a straightforward way. Nevertheless, the student may then consider the much more difficult problem of creating puzzles. One method is to adopt a trial-and-error process to fill the grid. Then numbers are removed with checking taking place that the uniqueness condition still holds by using the fast software tools that were created. These should be adequate to solve a position in a time of the order of milliseconds. In fact there are several authors who rely on automated systems to create puzzles and apply AI [9, 10].

The puzzle is interesting mathematically, and whilst the structure of the problem suggest the matrix be filled diagonally to start the creation process by initially filling the grid, a mathematics student may choose to take a closer look at this. Mathematically, the Sudoku puzzle is related to the branch of mathematics dealing with matrices and groups. Students with a mathematical background should be encouraged to investigate theoretical questions such as whether an initial configuration that can be shown by a computer analysis technique to have a single unique solution can always be logically solved.

This approach encourages students to look at the structure and principles of a game and let this guide a strategy. The puzzle is therefore rich in AI possibilities and can be an excellent AI testbed and a fine pedagogical tool. It clearly highlights the relationship between reasoning, pattern searching and memory. However, it is important to remember the puzzle represents a formal task and possibilities of generalisation to more complex computer games are limited. Many elements of complex computer games have mundane facets, a type of task that is not easily handled by the techniques we have been considering. Consider the much harder mundane task of completing a large jigsaw puzzle. The emphasis moves away from the reasoning that one would normally associate with human intelligence, to visualising and search. However the effort in creating an automated jigsaw solver is many orders of magnitude greater (though an interesting project nevertheless).

Sudoko does not have the necessary complexity of game space to test learning algorithms and heuristics. For this we must consider other games.

PEG SOLITAIRE

The game of peg or pegboard solitaire is frequently played on a wooden board with 32 marbles (or pegs) that sit in 32 recessed slots arranged in the form of a cross. The centre slot is normally unoccupied. The object of the game is to remove the marbles by jumping horizontally or vertically (removing the jumped marble) to leave one marble on the board. The game interface is easy to implement and is again a good student exercise. Fig. 4 shows a typical board where the programmer has chequered the board (though this has no effect on the play). This will be the starting point of any AI and the VB.NET 2003 implementation of this user interface is available from the author.

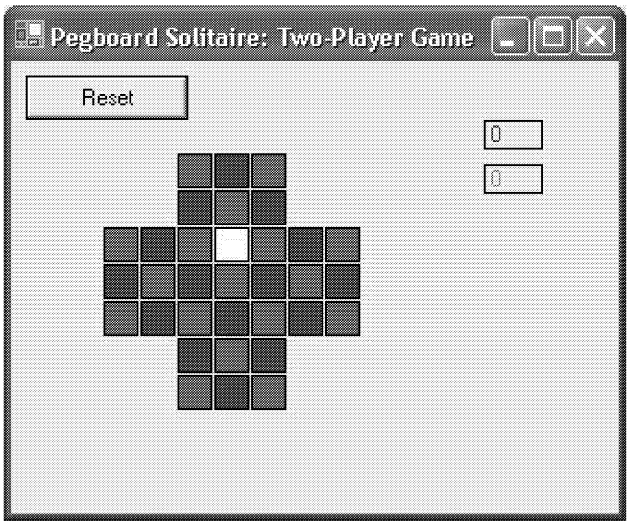


Figure 4: A representation of the game of pegboard solitaire (position of the initially empty square can be changed for variation)

The game is very much more difficult to solve using a computer than Sudoku because the complete tree has an estimated 10^{20} branches. It is therefore not possible to solve the game by a simple search technique. Nor is it possible to describe the strategy in clear formal terms. It is therefore necessary to apply heuristics (or rules of thumb) to narrow the search if this approach is followed. For example, it is known that it is easier to solve the puzzle if the block of remaining marbles is kept compact (Fig. 5). The high degree of symmetry can also be used to reduce unnecessary search. There is a significant theory on the game and some mathematical principles associated with the game can also be applied [11].

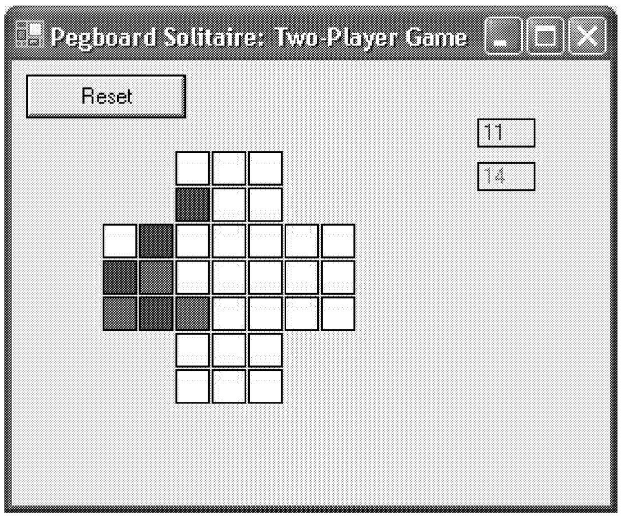


Figure 5: This game is completed with two successive jumps by the lower left purple counter, its immediate capture by the uppermost counter, and two horizontal jumps.

Students are asked to adapt the game for a 9 x 9 playing area for consistency with Sudoku (Fig. 6).

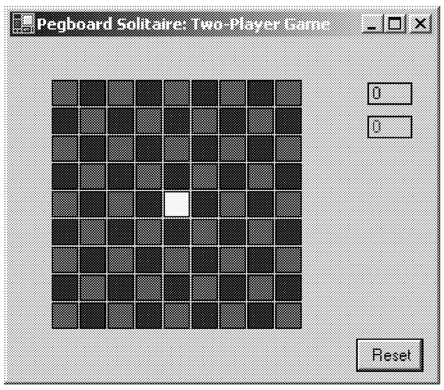


Figure 6: Games are altered for programming consistency as far as is possible. Here is Peg Solitaire playing space is made compatible with Sudoku and can be described by the same arrays. In this way generic objects may be developed.

Students have been encouraged to use neural networks to solve the puzzle, and evolutionary and genetic algorithms are readily applied. However the game lacks variety – the only real variable is the selection of the square that is initially empty. It is more interesting and challenging to create computer opponents than solvers. This means of course turning a two-play game into a single player game. Students are encouraged to apply game design skills to make the game more interesting by equipment and rule changes (though retaining the basic game principles). Students when asked to do this have demonstrated some ingenuity. One obvious modification is to recognise the connection with chequers (or draughts). If the 9 x 9 board is filled with alternatively coloured counters, the board is effectively two overlaid and offset grids, each player being confined to their own grid. This reduces the level of interaction below that which is desirable in a complex game. Nevertheless, games using this motif proved interesting.

In essence, this very simple game is a good trainer for more advanced AI principles and has the advantage that the methods are implemented with the minimum of programming through the simplicity of the game landscape.

CONCLUSION

The experience so far is that simple games are very effective at teaching foundational AI principles. Sudoku is effective at teaching about search and rule based systems; pegboard solitaire teaches, heuristics and learning. However, by the end of the process students are still very far away from being able to apply these ideas to the more complex arena of multi-player Internet computer games where the mundane aspects are significant and the constraints are poorly defined.

The very negative aspect of the approach was that students became very sceptical of what could be achieved with AI in any context because of the limited success of the techniques with even simple games. A very negative approach toward the application of AI to computer games became apparent with frequent criticism of the AI used in commercial games, and doubts as to whether it even constituted AI.

We considered this result undesirable and introduced an intermediate project-based module where it is necessary to learn strategies appropriate to the human mundane behaviour that is so common in game play but so difficult for the machine to deal with. For example, the game of Perudo has been found useful to explore these ideas and has been described elsewhere [12, 13, 14]. It is found that AI techniques can be relatively effective in this context and stimulates the student to apply AI to more complex computer games.

With this approach, games students have been found to emerge with a good understanding of AI and an awareness of the possibilities and its limitations when applied to complex systems.

REFERENCES

[1] A. Nareyek, “AI in computer games”, ACM Queue 1, **10** 58-65 (2004)

[2] G. A. Kaminka, M. M. Veloso, S. Schaffer, C. Sollitto, R. Adobbati, A. N. Marshall, A. Scholer, and S Tejada, “Gamebots: A flexible testbed for multiagent team research”, Communications of the ACM, **45**(1), 43 (2003).

[3] M. Gardner, “Tictacktoe and its complications" Scientific American, **225**, 102-104 (1971).

[4] M. Overmars, “Game design in education”, Technical Report UU-CS-2004-056 (2004).

[5] S. Singh, “The science of Su Doku”, BBC Focus, **155** (2005).

[6] H. Simonis, “Sudoku as a constraint problem”, www.icparc.ic.ac.uk/~hs/sudoku.pdf

[7] D. Eppstein, “Nonrepetitive paths and cycles in graphs with application to Sudoku”, Arxiv/cs.DS/0507053 (2005).

[8] A. Abdelilah, *Digital Circuits: Truth Tables, Minterms, Maxterms, Karnaugh Maps,* Engineer's Tutor Series, Weber System (1990).

[9] P. Sinden, “The little book of Sudoku”, Michael O'Mara Books Ltd (2005).

[10] For example, the forum http://act365.com/sudoku/ (Accessed 20 Sept 2005)

[11] A. B. Matos, “Depth-first search solves Peg Solitaire”, Technical Report, DCC-FC & LIACC, Universidade do Porto (1998).

[12] A. Macleod, “Perudo as a development platform for Artificial Intelligence”, 8*CGAIDE 2004 Conference on Computer Games: Artificial Intelligence, Design and Education*, Reading (2004).

[13] A. Macleod, “Selecting games for Artificial Intelligence”, *The Second Annual International Workshop on Game Design and Technology*, Liverpool John Moores University (2004).

[14] A. Macleod, “Game design through self-play experiments”, *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, Valencia (2005).

SITUATION SWITCHING IN THE AIBO ROBOT

Zhenke Yang
Leon Rothkrantz
Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science
Mekelweg 4,
2628 CD Delft
The Netherlands
{Z.Yang, L.J.M.Rothkrantz}@ewi.tudelft.nl

KEYWORDS

AIBO Entertainment robot, Situational AI, Reactive control, Clips, Robot behavior.

ABSTRACT

This work is focused on implementing situation switching in an AIBO robot inside an unpredictable environment. The goal is to achieve intelligent situation dependent reactive behavior without modifying the underlying planning algorithm. The focus is on visible intelligent behavior for a robot. Our ideas are illustrated using a benchmark experiment featuring an AIBO in a maze. The implementation is in accordance to the model and terminology for data fusion adopted by the Department of Defense Joint Directors of Laboratories.

1 Introduction

Robots are physical agents that perform tasks by manipulating the physical world. Traditionally, robots where applied in the manufacturing industry where they performed a single task in a fixed environment. For mobile robots in the entertainment industry however, parallel tasking in dynamic environments is desirable if not mandatory. Recent research has shown that, robot owners not only think of their robot as mobile, intelligent interfaces to information systems, but also expect their robots to behave similar to a familiar and amusing pet (Kobayashi et al. 2003, Matsui et. al. 1999). This implies that owners expect their robots to show some intelligent behavior. One manifestation of intelligent behavior is that the robot can act differently under different circumstances. For example, the robot walks in a straight line or circumvents depending on the existence of obstacles. In our view, sometimes changes in the environment are so subtle that they (1) require change in robot behavior to justify intelligence (externally) but (2) do not justify a change in the planning/problem solving algorithm (internal).

To illustrate this, consider a person driving to work. His goal is to arrive at the office on time. Depending on the current time and the current position the driver will have a different driving behavior. For example, if he has plenty of time he will adopt a relaxed driving style (and maybe allow the old lady and the children to cross the street first), otherwise he will adopt a more aggressive and risky driving style. Notice that the route the driver takes to get to work (the plan) does not change. In general, the driving style (actions taken by the driver) reflects the mental perception of the situation the driver thinks he is in. This mental perception of the situation depends on observables (e.g. time, place) and the current goal (e.g. get to work on time).

From an application development point of view, it is not desirable to have the designer of the planning algorithm to consider all possible situations. It is also not desirable to have the robot ignore the situations by showing the same behavior all the time. In this paper we propose a situation switching architecture as a solution to this problem. This paper describes the application of a situation switching architecture to solve a benchmark problem which consists of an AIBO robot exploring a maze.

2 Background

2.1 Robot planning and control

In the classical planning approach (Spalazzi 1998, Yang 1997), systems do not react to external events. Therefore success of a plan is not affected by changes in the outside world. When a failure occurs, a new plan is formed. Because of this, planning systems do not perform well in dynamic environments. An alternative approach is reactive control (Pearce et al. 1992, Safiotti 1993), which attempts to transform sensor data into information that directly affects the behavior of the robot. These systems usually do not have reasoning capabilities. Situated systems (Hanks et al. 1990, Saffiotti, et al. 1995) try to integrate the reasoning and reacting capabilities. An special kind of situated system is one with a reactive system (first layer) which interacts with a classical planner (second layer). Research on situated systems has focused on producing better plans. Our experiments, build on the results of this research, but our focus however is on visible intelligent behavior for a robot.

2.2 AIBO

The Sony AIBO entertainment robot was first released in June 1999. Due to (1) its wide array of sensors and activators, (2) the open architecture, which allows us to create behaviors for it and (3) its relatively low price, the AIBO is a good test subject for our experiments.

In our benchmark problem, the basic scenario involves a maze of about 3x3 meters build out of cardboard boxes. The AIBO is entered in this environment. The goal of the AIBO is to explore this maze and stop when it finds an exit sign (Figure 1). The AIBO has no prior knowledge about the size or shape of the maze. It only knows how to detect an exit sign and how to detect an obstacle. The AIBO must decide on which actions to perform to achieve its goal.

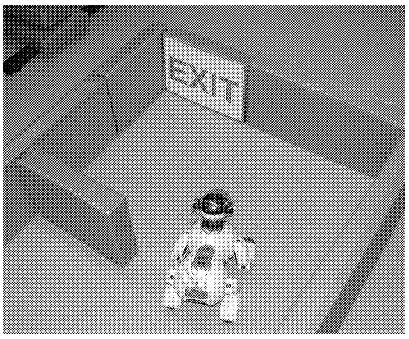


Figure 1: AIBO in Maze

Our goal is to let the AIBO show intelligent behavior by walking faster in some situations (e.g. through the long hallway) and slower in different situations (e.g. near corners) without modifying the exploration algorithm.

In the extended scenario there will also be an alarm sound. In this case the AIBO should walk carefully in all situations.

2.3 Multisensor data fusion

Multisensor data fusion seeks to combine data from multiple sensors to perform inferences that may not be possible from a single sensor alone. This is exactly the way we envision our AIBO to steer its behavior: by integrating data continually from different sensors to make inferences about the external world. As a result our implementation is done in accordance to the model and terminology for data fusion adopted by the Department of Defence Joint Directors of Laboratories (JDL) (Hall and Llinas 2001).

Following the JDL terminology, we use the term data fusion node. Data fusion nodes can be connected to each other so that the results of the processing in one node can be used as the input to the next node. A network of interconnected nodes thus formed is called a data fusion architecture.

To determine the distance and the location of an obstacle (if any), it is necessary to combine the sensor readings of the infrared distance sensor and the value of the neck joint. Since the AIBO is continuously scanning its environment by rotating its head left and right and the infrared distance sensor is located on the nose of the AIBO. The same holds for the detection of the exit sign using the camera (the camera is also located on the nose). Finally we can combine the location of an obstacle with the location of the exit sign to reinforce the belief that the exit sign has really been found (since the exit sign is a special obstacle).

3 Approach to the problem

Let us assume that the task the AIBO has to fulfill does not depend on the dynamics of the environment not due to the robot itself. This assumption implicates that the task can be solved using the classical planning approach. Let us furthermore assume, for simplicity, that the goal of the robot does not change during its lifetime. The above assumptions allow us the concentrate on the area we are actually interested in: the case where situations change (due to changes in the environment) within a task. A situation is a state of the environment which can be detected by the sensors of the robot. Situation changes are inherent to the dynamics of the environment.

Classical planning, including mechanisms for goal switching, is covered extensively in literature (Russel and Norvig 1995, Yang 1997). Context switching (the case where the goal as well as the situation changes simultaneously) falls within the area of operating systems. We will focus only on situation switching. The discussion above gives rise to Table 1.

Table 1: Focus Area

	Goal same	**Goal change**
Situation same		Task switch
Situation change	Situation switch	Context switch

3.1 Situation Estimation and Selection

The AIBO is equipped with a range of sensors to monitor the dynamics in its environment. These observations determine the state of the AIBO's internal representation of the environment. We assume that knowledge of the state at time t is sufficient for the AIBO to determine the situation at time t i.e. no historical values are necessary. Of course there are issues with sensors: noise, situation constituted by the sensor readings can be ambiguous, etc. But these issues are left out of the scope of this paper.

With the above assumptions, situation estimation can be reduced to the process by which the data fusion architecture is used to transform the sensor readings (observations) in to a state of the environment and situation selection is a function that maps a given state into a set of situations. Of course, more sophisticated and versatile situation estimation and selection methods could be devised. However, this method of situation selection is sufficient for our problem.

3.2 Skills-sets

The set of actions a robot can perform are called the skills of the robot. Skills form the robot-specific interface with the world, in the sense that a skill defines how a higher level command is transformed into continuous control of the robots actuators. Generally robots can vary greatly in physical characteristics and sensors capabilities. As a result, the skill of each robot can also vary greatly between robots and environments. This concept is not new, Skill have been developed for various robots and various environments (Bonasso et al. 1995, Slack 1992).

As every situation essentially constitutes a different environment in which the robot has to perform its tasks, each situation may demand different skills. We therefore create a set of skills and let the robot choose the appropriate skill to use depending on the situation it is in. As a result, the situation switching process is essentially reduced to the selection of the appropriate skill from the skill-set given the current situation.

3.3 Selection of skill set from situation

The robot behavior is defined by task-directed selection of a skill from the skill-set. Using a slight modification of universal plans (Schoppers 1987), selection rules for skill sets, in a given situation, can be expressed as follows:

If a situation satisfying condition P arises while trying to achieve goal G, then use skill set S to perform the actions.

As an illustrative example consider an environment with 2 possible situations: Hazardous (SH) and Save (SS). The possible actions are moving forward (Fx) and Turning (Tx) where x denotes the speed and the degree respectively. A skill for situation SS = {F40, T90} and a skill for SH = {F10, T20}. As a consequence, an environment satisfying situation SS will cause the robot to move and turn faster then an environment satisfying situation SH.

4 Implementation

Our prototype is based on a client-server architecture, with the server being the AIBO and the client being a PC. The data processing (situation detection and switching, exit detection, exploration etc.) are done on the PC and AIBO is responsible for sending raw sensor information from the sensors and executing the commands send from the client. The communication between client and server is through WiFi. Our long term goal is to have both client and server running on AIBO and in this way having an entirely autonomous robot.

4.1 AIBO

For this experiment we created a combined memory stick containing both URBI (Baillie 2005) and Tekkotsu (http://www-2.cs.cmu.edu/~tekkotsu/) software.

Tekkotsu provides walking routines (that were not available for URBI at the time the experiment was done) while URBI provides better control over and easier access to the sensors. In order to prevent conflicting/simultaneous accessing of the AIBO hardware the actuator functions in URBI have been disabled in favor of Tekkotsu. URBI is mainly used to retrieve sensor information: Distance IR sensor, joint values, camera images, microphone data. Tekkotsu is used for walking and lower resolution UDP video from the camera.

4.2 PC client

The client is implemented according to the JDL data fusion model. The client takes care of the retrieval and distribution of the sensor data from the AIBO (SensorManager in Figure 2). Each data fusion node can get all the sensor information available by subscribing to the appropriate services provided by the SensorManager. The client also takes care of the translation from a symbolic commands used in the skill sets to concrete AIBO understandable commands (ActionManager in Figure 2). In addition the client consists of several modules which are implemented as data fusions nodes.

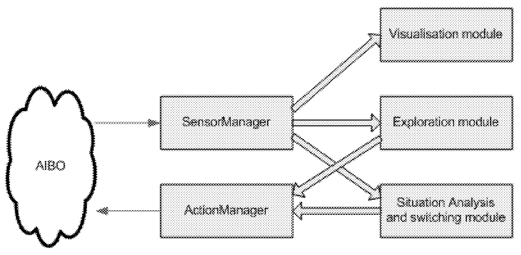


Figure 2: Module Overview

The Visualization module

This module is intended for global control of the application and to show feedback to the user. Included are views for real-time camera images and a representation of the current map created by the exploration module (Figure 3). These views are provided for convenience and have no functional meaning in our experiment.

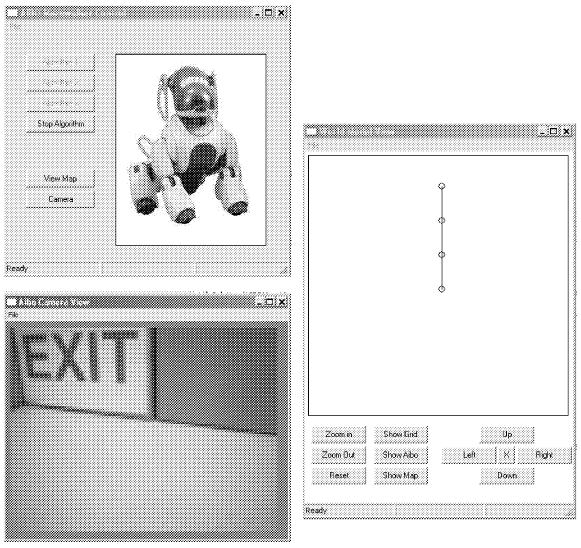


Figure 3: Views of the Visualization Modules

The Exploration module

The exploration module constitutes the planning part of the system (in the classical sense). Since planning is not the focus of our research, the exploration algorithm is kept extremely simple. Listing 1 shows the pseudo code for the exploration algorithm.

```
exitfound = false;
while (not exitfound)
   /* Get distance to obstacle in front.*/
   dc = GetIRDistanceChest();
   if (dc > THRESHHOLD)
      /* walk save distance forward */
      Forward(0.5*THRESHHOLD);
   else
      Turn90Degrees();
   fi
   updateMap();
   exitfound = detectExitSign();
End While
```

Listing 1: The Exploration Algorithm

Basically, the AIBO walks a save distance forward if there is no obstacle in front it, otherwise it makes a 90 degrees turn to the left. Until the exit sign is detected.

The situation analysis and switching module

Given the nature by which the situation switching conditions are expressed, the choice to use a production system, such as CLIPS, to implement the situation analysis module is a logical one. In addition, the CLIPS source code is already ported to an OPEN-R object (which can be run directly on a AIBO). So this also fits well with our long term goal of complete autonomy.

In our scenario we distinguish 4 different situations: {normal, alarm, clear, cornered} and 3 different skill sets labeled {Normal, Cautious, Fast, Alarm}. Table 2 shows the relationship between the situations and the skill-sets.

Table 2: conditions for the situations and mapping to skill set

Situation	Condition	Skill set
Normal	Default	Normal
Alarm	If alarm sound is detected	Cautious
Clear	If distance in front is greater then Threshold and NOT Alarm	Fast
Cornered	If distance in front is less then Threshold and NOT Alarm	Cautious

5 Evaluation

The benchmark problem described in this paper is a first experiment to show our ideas. The working system was demonstrated on several occasions. As the focus of the demonstrations was on other research areas (path finding, exit sign/landmark recognition, etc.) the situation switching part discussed in this paper was mostly lost to the public.

Currently work is done to extend this problem with the sound modality. In the extended problem, sound is used to influence the situation. For example, the AIBO has to walk hastily towards the source of the sound when an urgent sound is played. With this extension, situation switching as well as the data fusion model can be further evaluated.

The architecture used did not modify in any substantial way the task of the planner. Moreover, the classical planning approach can be considered a special case of our architecture, namely the case of that of one situation and one set of skills. As a result, dividing situation switching and task switching as proposed in this paper, is not difficult to achieve and can add another dimension in robot/character behavior in games.

6 Future Work

A situation can seldom be determined by observations at a single point in time. A conclusive statement about the situation, usually involves an analysis of a series of related observations at several distinct moments in time. Therefore situation switches involve keeping track of sensor history. Furthermore, a situation is seldom determined by readings from a single sensor (as is the case in our benchmark example). Invariably, it takes a combination of sensor information from different modalities to determine the situation. In this case data fusion techniques (e.g. Kalman filtering) can be used to determine the current situation and predict the future situation. Finally, in our benchmark problem, a simple rule-based selection of skills is implemented. Next, other AI techniques (genetic algorithms, reinforcement learning) can be used to create learned skill-sets and selection rules.

7 Conclusions

This paper presents a solution and a proof of concept to the problem of achieving an integration of planning and sensor driven reaction. Our idea is realized by a benchmark problem featuring an AIBO in a maze. Previous research on situated control has focused on producing better plans. Our experiments build on the results of this research, but the focus is on immediately visible intelligent behavior. Dividing situation switching and task switching in games as proposed in this paper, is not difficult to do and can add another dimension in robot/character behavior in games. The ideas presented in this paper can readily be adopted in games.

References

Baillie, J., "URBI Server for AIBO ERS2xx ERS7 Introduction Manual v1.0", January 2005. (http://www.urbiforge.com/eng/)

Bonasso, P et. al., "Experiences with an Architecture for Intelligent, Reactive Agents", 1995

Hall, D. L., Llinas, J., "Handbook of Multisensor Data Fusion," CRC Press, 2001.

Hanks S. et. al., "Issues and Architectures for Planning and Execution. In Proceedings of the workshop on Innovative Approaches to Planning, Scheduling and Control", pages 59-70, 1990.

Kobayashi, A. et. al., "Robot Middleware Architecture Mediating Familiarity-Oriented and Environment-Oriented Behaviors". in the 5th IEEE International Symposium on Computational Intelligence in Robotics and Automation, pp.544-551, July 2003

Matsui, T. et. al., "Integrated Natural Spoken Dialogue System of Jijo-2 Mobile Robot for Office Services". In Proc. Of the 16th National Conference on Artificial Intelligence (AAAI-99), Florida, July 1999.

Pearce M. et. al., "The Learning of Reactive Control Parameters Through Genetic Algorithms". IEEE/RSJ International Conference on Intelligent Robots and Systems, 1992 .

Russel S. and Norvig, P., "Artificial Intelligence – A Modern Approach", Prentice Hall, Englewood Cliffs, New Jersey, 1995.

Saffiotti, A. et. al., "A Multivalued Logic Approach to Integrating Planning and Control". Artificial Intelligence, 76(1-2):481-526, 1995.

Saffiotti, A., "Some Notes on the Integration of Planning and Reactivity in Autonomous Mobile Robots", 1993. AAAI Spring Symposium on Foundations of Planning, pp. 122-126. March 1993.

Slack, M. G., "Sequencing Formally defined reactions for robotic activity: Integrating raps and gapps", In Proceedings of SPIE's Conference on Sensor Fusion, 1992.

Schoppers, M. J., "Universal plans for Reactive Robots in Unpredictable Environments". In Proc 10th IJCAI, 1987

Spalazzi, L., "An Architecture for planning in embedded systems". Istituto di Informatica, University of Ancone, Italy, 1998

Yang, Q., "Intelligent Planning: A Decomposition and Abstraction Based Approach". Springer Verlag, Berlin, Germany, 1997.

SYNTHETIC CHARACTERS AND AGENTS

Advanced Synthetic Characters, Evil, and E*

Selmer Bringsjord[1], Sangeet Khemlani[2], Konstantine Arkoudas[3], Chris McEvoy[4], Marc Destefano[5], Matthew Daigle[6]
Department of Cognitive Science$^{1-5}$
Department of Computer Science^{1,3,4}
Rensselaer AI & Reasoning Laboratory:$^{1-5}$
http://www.cogsci.rpi.edu/research/rair/index.php
Rensselaer Polytechnic Institute (RPI)
Troy NY 12180 USA
{selmer,arkouk,mcevoc,khemls,destem}@rpi.edu
6: Dept. of Computer Science Vanderbilt University Nashville TN mdaigle@isis.vanderbilt.edu

Abstract

We describe our approach to building advanced synthetic characters, within the paradigm of logic-based AI. Such characters don't merely evoke *beliefs* that they have various mental properties; rather, they must actually *have* such properties. You might (e.g.) believe a standard synthetic character to be evil, but you would of course be wrong. An *advanced* synthetic character, however, can literally *be* evil, because it has the requisite desires, beliefs, and cognitive powers. Our approach is based on our RASCALS architecture, which uses simple logical systems (first-order ones) for low-level (perception & action) and mid-level cognition, and advanced logical systems (e.g., epistemic and deontic logics) for more abstract cognition. To focus our approach herein, we provide a glimpse of our attempt to bring to life one particular advanced synthetic character from the "dark side" — the evil character known simply as E. Building E entails that, among other things, we formulate an underlying logico-mathematical definition of evil, and that we manage to engineer both an appropriate presentation of E, and communication between E and humans. For presentation, which we only encapsulate here, we use several techniques, including muscle simulation in graphics hardware and approximation of subsurface scattering. For communication, we use our own new "proof-based" approach to Natural Language Generation (NLG). We provide an account of this approach.

The Dearth of AI in AI

There's an unkind joke — which made the rounds (e.g.) at the Fall 2004 AAAI Fall Symposium on Human-Level AI — about the need to create, within AI, a special interest group called 'AI'. This kind of cynicism springs from the not uncommon, and not totally inaccurate, perception that most of AI research is aimed at exceedingly narrow problems light years away from the cognitive capacities that distinguish human persons.[1]

Human-level AI is now so unusual that an entire upcoming issue of *AI Magazine* will be devoted to the subject — a bit odd, given that, at least when the field was young, AI's journal of record would have *routinely* carried papers on mechanizing aspects of human-level cognition. Seminal AI thinkers like Simon, Newell, Turing — these researchers didn't shy away from fighting to capture human-level intelligence in machine terms. But now their attitude seems moribund.

But gaming, simulation, and digital entertainment (and hereafter we refer simply to 'gaming' to cover this entire field/market), thankfully, are different: *ultimately* anyway, they call for at least the *appearance* of human-level AI (Bringsjord 2001). (On a case-by-case basis, as various games show (e.g., *The Sims* (Electronic Arts Inc. 2000)), a *non*-advanced character will of course do just fine.) Gaming doesn't strive just for a better SAT-based planner, or another tweak in a learning algorithm that doesn't relate in the least to human learning. A SAT planner doesn't constitute a virtual person. But that's precisely what we want in gaming, at least ultimately. And even in the short term we want characters that at least *seem* human. Methodologically speaking, gaming's best bet for characters that seem human is to bite the bullet and strive to engineer characters that have what it takes to *be* human. This, at least, is our strategy.

Gaming and Full-Blown Personhood

Now, there are various ways to get clearer about what gaming, at least in the long-term, needs when it comes to human-level intelligence. One way is to say simply that gaming needs artificial creatures which, behaviorally at any rate, satisfy one or more plausible proposed definitions of personhood in the literature. One such definition has been proposed by Bringsjord in (Bringsjord 1997). This definition essentially amounts to the view that x is a person if and only if x has the *capacity*

1. to "will," to make choices and decisions, set plans and projects — autonomously;
2. for consciousness, for experiencing pain and sorrow and happiness, and a thousand other emotions — love, passion, gratitude, and so on;
3. for *self*-consciousness, for being aware of his/her states of mind, inclinations, preferences, etc., and for grasping the concept of him/herself;
4. to communicate through a language;
5. to know things and believe things, and to believe things about what others believe, and to believe things about what others believe about one's beliefs (and so on);
6. to desire not only particular objects and events, but also changes in his or her character;

*The R&D described in this paper has been supported in part by much appreciated grants from AFRL-Rome and DARPA-IPTO.

[1] An endless source of confirming examples can be found in the pages of the *Machine Learning* journal. The dominant learning technique that you yourself employ in striving to learn is *reading*; witness what you're doing at the moment. Yet, a vanishingly small amount of R&D on learning is devoted to getting a computer program to learn by reading.

7. to reason (for example, in the fashion exhibited in the writing and reading of this very paper).

Unfortunately, this list is daunting, especially if, like us, you really and truly want to engineer a virtual person in the *short term*. A large part of the problem is consciousness, which we still don't know how to represent in third-person machine terms (Bringsjord 1998; Bringsjord 2001). But even if we leave aside consciousness, the rest of the attributes in the above list make for mighty tough challenges. In the section *"Making the Challenge of Personhood Tractable"* we shall retreat from this list to someting doable in the near term, guided by particular scenarios that make natural homes for E. But in the end, whatever appears on this list is an engineering target for us; in the long term we must confront each clause. Accordingly, in the section *"How Does E Talk?"* we explain how we are shooting for clause 4, communication. We have made progress on some of the other clauses, but there is insufficient space to present that progress herein. Clause 5 is one we believe we have pretty much satisfied, via the formalization and implementation given in (Arkoudas & Bringsjord 2005).[2]

Current State of the Art versus Computational Persons

Synthetic Characters in Gaming

What's being done now in gaming, relative to full-blown personhood, is clearly inadequate; this can be quickly seen by turning to some standard work: Figure 1 shows an array of synthetic characters from the gaming domain; these will be familiar to many readers.[3]

None of these creatures has anything close to the distinguishing features of personhood. Sustained treatments of synthetic characters and how to build them are similarly limited. For example, consider Figure 2, taken from (Champandard 2003).[4] As a mere FSA, there is no knowledge and belief, no reasoning, no declarative memories, and no linguistic capacity. In short, and this is perhaps a better way of putting the overall problem infecting todays's virtual characters, all of the cognitive capacities that distinguish human persons, according to the science of cognition (e.g., (Goldstein 2005)), are missing. Even the state of the art using cognitive architectures (e.g., SOAR) is primitive when stacked against full-blown personhood (Ritter *et al.* June 2002).

[2]A preprint is available online at http://kryten.mm.rpi.edu/arkoudas.bringsjord.clima.crc.pdf.

[3]Worst to best, in our eyes: Top-left, The Legend of Zelda; SC spits text upon entering room. Top-right, Chrono Trigger; tree-branching conversations. Middle-left, Might & Magic VI (Shopkeepers). Middle-right, Superfly Johnson from Daikatana; behavior scripting, attempts to follow player and act as a sidekick (fails!). Bottom-left, Galatea – Interactive Fiction award winner for Best NPC of 2000 (text-based). Bottom-right, Sims 2. But even here, nothing like what our RASCALS architecture has is present.

[4]This is an excellent book, and it's used in our lab for building synthetic characters. But relative to the loftier goals of reaching *bona fide* personhood in artificial characters, there's clearly a lot of work to be done.

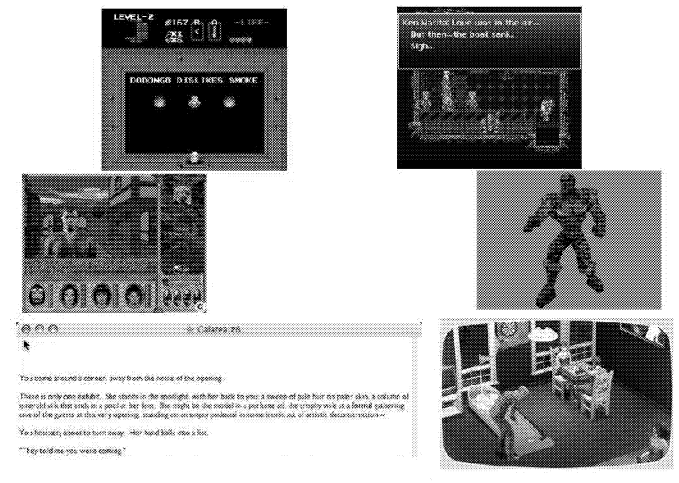


Figure 1: Sample Synthetic Characters

What About Synthetic Characters in Cutting Edge Research?

What about research-grade work on synthetic characters? Many researchers are working on synthetic characters, and have produced some truly impressive systems. However, all such systems, however much they *appear* to be human persons, aren't. We now consider three examples of such work, and show in each that the character architectures don't have the underlying cognitive content that is necessary for personhood.

REA

An agent developed by (Cassell *et al.* 1999) known as REA is an example of a successful, robust agent whose developers focused primarily on embodied conversation and the conversational interface. She is described as being an expert in the domain of real estate, and interactions with REA are both believable and informative.

REA, however, is representative of many of the industry's most successful agents in that she excels at content management, but fails to deliver rich emotive and cognitive functionality. REA, after all, cannot generate English from *arbitrary* underlying knowledge. Like many of her peers, REA's underlying cognitive capabilities are modeled in an ad-hoc fashion. Her personality is in no way defined; her interactions within a particular situation lack subtlety and depth. While she excels as a simulated character and a conversational agent, she is bereft of the rich cognitive content with which advanced synthetic characters must behave.

The BEAT Architecture

In an engaging paper (Gratch *et al.* 2002), Gratch and colleagues present an architecture for developing rich synthetic characters. This architecture is known as the Behavior Expression Animation Toolkit Text-to-Nonverbal Behavior Module (BEAT). Under this architecture, emotion and cognitive content are produced systematically in a *simulation-based* approach.

Their simulation-based approach is built on top of *ap-*

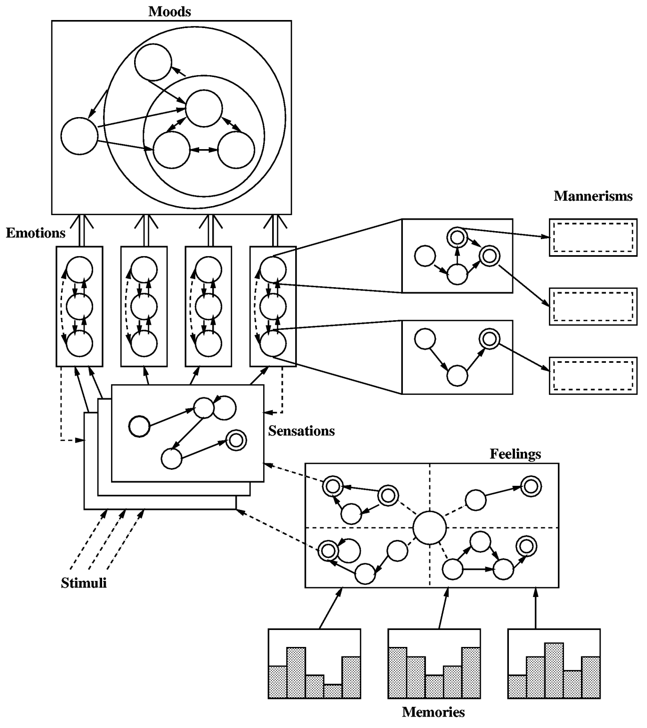


Figure 2: Impoverished Formalism for Synthetic Characters

praisal theories of emotion, where emotions emerge from analysis of events and objects in a particular domain with respect to the agent's goals, standards, and attitudes. But as Gratch et al themselves point out, appraisal theories "are rather vague about the assessment process...A promising line of research is integrating AI-based planning approaches, which might lead to a concretization of such theories." We will present the RASCALS paradigm as one that utilizes precisely the AI-based planning techniques Gratch et al. regard as promising.

Unfortunately, while Gratch and colleagues make wonderful advancements in the logistics of realizing agents, the issue of developing rich underlying cognitive content is eschewed. Even assuming that their simulation-based approach utilizes robust AI-based planning, the focus is not on developing true cognitive content but rather on its simulation and modeling.

Believable Interactive Embodied Agents

An approach more focused on building believable characters was proposed by (Pelachaud & Poggi 2002). They argue that research should include three distinct phases:

- *Phase 1: Empirical Research.* This phase involves research "aimed at finding out the regularities in the mind and behavior of Human Agents, and at constructing models of them."
- *Phase 2: Modeling Believable Interactive Embodied Agents.* Here, "rules are formalized, represented, and implemented in the construction of Agents."
- *Phase 3: Evaluation.* Finally, agents are tested on several levels, including "how well they fit the User's needs and how similar they look to a real Human Agent."

The "rule formalization" characterized in Phase 2 is, as Pelachaud and Poggi point out, indispensable when building believable characters. Since such rule formalizations are all expressible in first-order logic, their approach is actually a proper subset of the RASCALS approach. But formalizing and implementing rules is not enough to achieve true cognition; after all, cognition involves much more than simple rules/first-order logic. Iterated beliefs are beyond the reach of first-order logic. Finally, while Pelachaud and Poggi elaborate on linguistic rules and formalizations, they fail to mention anything about modeling cognition or interacting with a given knowledge base, and they make no remarks concerning the logistics behind rule formalization and implementation. The agents described therein all possess rudimentary cognitive content but come nowhere close to true cognitive or emotive capacity.

Making the Challenge of Personhood Tractable

How can we make the challenge of engineering a virtual person tractable in the very short term? Our lab has a two-part answer. First, assimilate everything out there regarding the *craft* of making viewers and users *believe* that the synthetic character they interact with is a genuine person. This is the same route that was followed by Bringsjord and Ferrucci in the design of the BRUTUS story generation system (Bringsjord & Ferrucci 2000). In a nutshell, B&F studied the literature on what responses are desired in readers by clever authors, and then reverse engineered back from these responses to a story generation system that triggers some of them. In connection with synthetic characters, this general strategy has impelled us to build up a large library on the design of synthetic charaters in stories and movies. In addition, we have built up a library of characters in film — specifically one that specializes in candidates for true evil. Within the space we have herein, however, this general strategy, and the results so far obtained, can't be presented. So we will settle here for a shortcut; it's the second part of our two-part answer. The shortcut is to work from concrete scenarios backwards by reverse engineering. We currently have two detailed scenarios under development. One is based on the evil people whose personalities are revealed in conversations in (Peck 1983); we leave this one aside for now. The second scenario, which is part of R&D undertaken in the area of wargaming, can be summarized as follows. (At the conference, we would provide a demo of conversation with E regarding the first of these scenarios, where that conversation conforms to our account of evil; see *On our Formal Account of Evil.*)

E in Scenario 2, and Inference Therefrom

Let us imagine a man named simply E, a brutal warlord in a war-torn country. E is someone you're going to have to vanquish. He has moved up the ranks of the underworld in post-apocalyptic America after "success" in many, many murderous missions. E has taken a number of prisoners from an organization (let's call it simply *O*) he seeks to intimidate. *O* is chosen specifically because it is trying to rebuild the fractured US in the direction of a new federal governing[5]. Conforming to what has unfortunately become a gruesome pattern, E decides to film the beheading of one of these poor prisoners, and to release the video to *O*.

[5]Coincidentally, we have recently learned that the game *Shattered World* for the XBox is related to our scenario.

Given just this small amount of information, what can we infer about E's knowledge and reasoning? That it has at least the following six attributes:

1. *Mixed Representation.* E's knowledge is not simply linguistic or symbolic in nature. It includes visual or pictorial knowledge as well. For example, E clearly is thinking in terms of mental images, because he plans to gain leverage from the release of images and video. In addition, though it isn't pleasant to contemplate, E certainly has a "mental movie" that he knows he can turn into real life: he envisions how such executions work before performing them.
2. *Tapestried.* Presumably E's knowledge of his prisoners is relatively new. But this new knowledge is woven together with extensive prior knowledge and belief. For example, in E's case, he has extensive knowledge of O, and its principles regarding treatment of prisoners.
3. *Extreme Expressivity.* E's knowledge and reasoning requires highly expressive propositions. For example, he believes that O believes that it is universally forbidden to execute prisoners, and he believes that some of those aiding the United States' rebuilding effort will be struck with fear once the execution is complete and suitably publicized, and that that fear will affect their beliefs about what they should and shouldn't do.
4. *Mixed Inference Types.* E's reasoning is based not only on deductive inference, but also on educated guesses (abduction), and probabilistic inference (induction).
5. *Uses Natural Language.* E communicates in natural language, with his comrades, and with others as well.
6. *Multi-Agent Reasoning.* E is of course working in coordinated fashion with a number of accomplices, and to be effective, they must reason well as a group.

Working within the paradigm of logic-based AI (Bringsjord & Ferrucci 1998a; Bringsjord & Ferrucci 1998b; Nilsson 1991; Genesereth & Nilsson 1987), and using the MARMML knowledge representation and reasoning system, which is based on: the theory known as mental metalogic (Yang & Johnson-Laird 2000a; Yang & Johnson-Laird 2000b; Yang & Bringsjord 2005; Rinella, Bringsjord, & Yang 2001; Yang & Bringsjord 2001a; Yang & Bringsjord 2001b; Yang, Braine, & O'Brien 1998), the Denotational Proof Language known as Athena (Arkoudas 2000), Barwisean grids for diagrammatic knowledge and reasoning (see the mathematical section of (Barwise & Etchemendy 1995)), and RASCALS[6](see Figure 3), a revolutionary architecture for synthetic characters, we are building a virtual version of E that has the six attributes above.

Brief Remarks on the RASCALS Architecture

Let us say a few words about RASCALS, a brand new entry in the field of compuational cognitive modeling, which revolves around what are called *cognitive architectures* (e.g., SOAR (Rosenbloom, Laird, & Newell 1993); ACT-R (Anderson 1993; Anderson & Lebiere 1998; Anderson & Lebiere 2003); CLARION (Sun 2001); Polyscheme (Cassimatis 2002; Cassimatis *et al.* 2004)). What makes the RASCALS cognitive architecture distinctive? There is insufficient space here to convey any technical detail (for more details, see (Bringsjord forthcoming)); we make just three quick points about RASCALS, to wit:

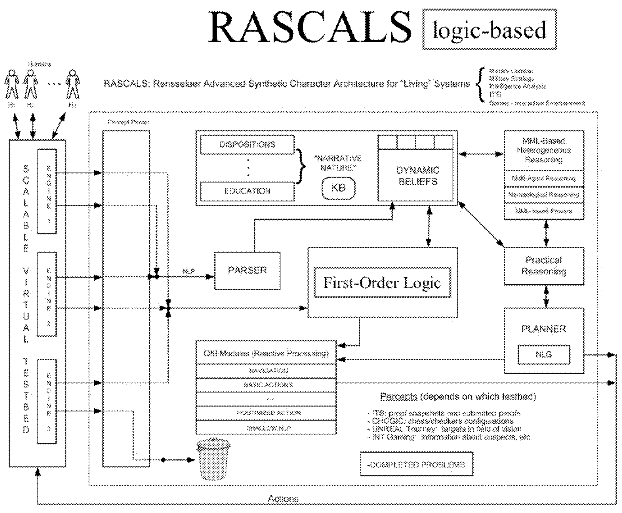


Figure 3: RASCALS: **R**ensselaer **A**dvanced **S**ynthetic **C**haracter **A**rchitecture for **L**ogical **S**ystems

- All other cognitive architectures we know of fall far short of the expressive power of RASCALS. For example, SOAR and ACT-R struggle to represent (let alone reason quickly over) textbook problems in logic (e.g., the Wise Man Problem = WMP) but in RASCALS such representations are effortless (see (Arkoudas & Bringsjord 2005) for the solution to WMP in Athena, included in RASCALS).
- The great challenge driving the field of computational cognitive modeling (CCM) is to unify *all* of human cognition; this challenge can be traced back to the birth of CCM in the work of Newell 1973. Such unification is achieved in one fell swoop by RASCALS, because *all* of cognition can be formalized and mechanized in logic (though doing so requires some very complicated logics well beyond first-order logic, as in (Arkoudas & Bringsjord 2005)).[7]
- While logic has been criticized as too slow for real-time perception-and-action-heavy computation, as you might see in first-person shooter (as opposed to a strategy game, which for obvious reasons fits nicely with the paradigm of logic-based AI), it has been shown that RASCALS is so fast that it can enable the real-time behavior of a mobile robot. We have shown this by having a logic-based mobile robot successfully navigate the wumpus world game, a

[6]**R**ensselaer **A**dvanced **S**ynthetic **C**haracter **A**rchitecture for **L**ogical **S**ystems

[7]It will naturally occur to some skeptics to enquire about traditional-style learning, and speech recognition. As to the former, it's well-known that there are logic-based approaches to divining a function f by repeated trial; see, e.g., (Russell & Norvig 2002). There are also well-known knowledge-based (which become, in RASCALS, more formal, logic-based) techniques for learning: EBL, RBL, etc.; again, see (Russell & Norvig 2002) for a survey. Of course, RASCALS does reject purely statistical and probabilistic approaches to learning (and other cognitive phenomena). That seems quite unsurprising, since statistical approaches in AI routinely reject, to their peril, declarative/logic-based techniques. As to the latter problem, RASCALS insists that all language be represented in logical form, and Bringsjord concedes that this is currently not achieved, nor even on the horizen. However, with respect to natural language understanding, all researchers, whatever their approach, are currently in the same dismal boat.

staple in AI. (See Figures 4 and 5.)

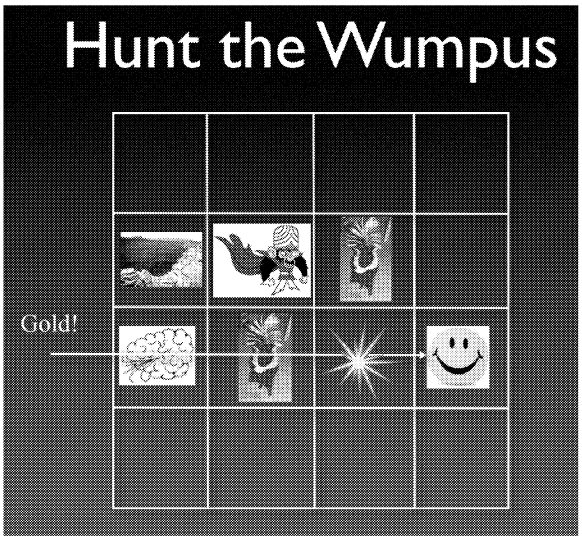


Figure 4: The Wumpus World Game

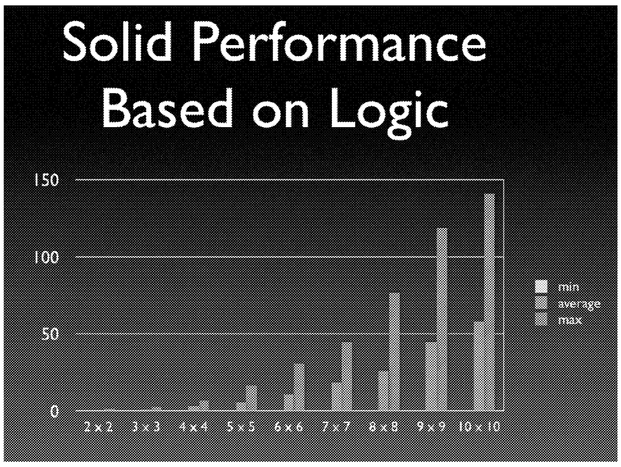


Figure 5: Performance of a RASCALS-Powered Robot in the Wumpus World

To show part of the underlying structure of E in connection with the attribute *Extreme Expressivity*, we now present an informal version of the formal account of evil that is implemented in our RASCALS architecture. This account specifically requires logics expressive enough to handle knowledge, belief, and ethical concepts. These logics go well beyond first-order logic; details and an implementation can be found in (Arkoudas & Bringsjord 2005). In the section *"E: The Presentation Level"* we explain the technology that allows E to speak naturally in English; that is, we show there part of the underlying structure of E associated with *Uses Natural Language*.

On our Formal Account of Evil

If we charitably push things in the direction of formally representing a definition of evil,[8] then we can understand Feinberg 2003 as advancing pretty much this definition:

Def 1 Person s is evil iff there exists some action a[9] such that

1. performing a is morally wrong;
2. s is morally blameworthy for performing a;
3. s's performing a causes considerable harm to others; and
4. the reasons or motives for s's performing a, along with "the elements that ground her moral blameworthiness," are unintelligible.

This is a decent starting place, but for us there are problems. For example, imagine that E invariably fails to cause *actual* harm. Surely he would still qualify as evil even if he were a bumbling villain. (If the knife slipped when he attempted decapitation, he would still be just as black-hearted.) This means that clause 3 should at least be replaced by

$3'$. s performs a in the hopes of causing considerable harm to others

But even this new definition, for reasons we don't have space to explain, is wholly inadequate. To give just a flavor for what E is currently based upon, we present simply our current best replacement for clause 4:

$4''$ were s a willing and open participant in the analysis of reasons and motives for s's seeking to perform a, it would be revealed that either

(i) these reasons and motives are unintelligible, or

(ii) s seeks to perform a in the service of goal g, and

(a) the anticipatable side-effects e of performing a are bad, but s cannot grasp this, or

(b) g itself is appraised as good by s when it is in fact bad.

Just this clause alone required much sustained analysis. (For a full chronicle of the evolution of a formally refined definition of betrayal from a rough starting one, see the chapter "Betrayal" in (Bringsjord & Ferrucci 2000).)

Keep in mind that this is still informal, kept that way in the interests of easing exposition. In the RASCALS-based implementation of E, evil must be expressed in purely formal form, which requires, again, that we use advanced logics of belief, knowledge, and obligation.[10]

Keep in mind as well that we're not claiming that we have the perfect definition of evil. Some may object to our definition, and some of their objections may be trenchant. But the important point is to see how rich evil is — to see that it involves all kinds of highly cognitive powers and concepts that simply aren't found in today's synthetic characters. To be evil, one has to have beliefs, desires, and one has to have a lot of knowledge. The detailed configuration of these elements may not be exactly as we claim they ought to be, but no one can deny that the elements are needed. Without them, a synthetic character who is supposed to be evil is only a fake shell. And in the end, the shell will be revealed to be a shell: the illusion, at some point, will break down.

[8]Feinberg's work is informal, and not suitable for direct use in AI and computer science.

[9]Or omission.

[10]For a look at the deontic logic (i.e., the logic of ethical concepts) we are relying upon, see (Horty 2001). Our mechanization of this this logic will be presented at the AAAI November 2005 Fall Symposium on Machine Ethics. The paper is available online at http://kryten.mm.rpi.edu/FS605ArkoudasAndBringsjord.pdf.

How Does E Talk?

As everyone knows, once the daunting challenge of rendering consciousness in computational terms is put aside, the greatest remaining challenge is that of giving an advanced synthetic character the power to communicate in a natural language (English, French, etc.) at the level of a human person. As you'll recall, communicative capacity is one of the clauses in the definition of personhood presented above. A plausible synthetic character must necessarily communicate in a fluid, robust manner. How, then, is such a rich form of communication implemented in E?

Reconciling Knowledge Representation and NLG

E speaks by parsing and processing formal knowledge; he develops an ontology based on internal and external queries, and then reasons over his knowledge to produce meaningful content. This content is then sent to his NLG module, translated into English, and finally presented to the user. Before we examine what goes on inside E's NLG module, let's take a moment to examine how E produces "meaningful content."

When we ask E a question, we are clearly interested in an answer that is both relevant and meaningful, an answer indistinguishable from those given by a real person. Assuming we have incomplete knowledge, suppose we ask of E, "Is John dangerous?" E approaches this question through formal logical analysis. The idea is to have E determine incontrovertibly whether John is dangerous or not. So, for instance, suppose E's knowledge base includes the following three facts:

1. DANGEROUS PEOPLE HAVE AUTOMATIC WEAPONS.
2. JOHN HAS A BERETTA AR-70 ASSAULT RIFLE.
3. THE BERETTA AR-70 ASSAULT RIFLE IS AN AUTOMATIC WEAPON.

None of the information above explicitly tells E whether John is dangerous or not, but clearly, when presented the above query, we want E to answer with an emphatic "Yes." Still, the answer itself is not enough. To ensure that E understands the nature of the question as well as the information he is dealing with, he must, upon request, provide *a justification for every answer*. The justification presented to the user is a formal proof, translated into English. Thus, E could answer as follows:

JOHN IS IN FACT DANGEROUS BECAUSE HE HAS A BERETTA AR-70 ASSAULT RIFLE. SINCE A BERETTA AR-70 ASSAULT RIFLE IS AN AUTOMATIC WEAPON, AND SINCE DANGEROUS PEOPLE HAVE AUTOMATIC WEAPONS, IT FOLLOWS THAT JOHN IS DANGEROUS.

Content is thus generated in the form of a formal proof. In general, the proofs generated will be more complex (they will use larger knowledge bases) and more sophisticated (they will use deontic and epistemic logic).

While the example is simple and rudimentary (that is, it makes use of only first-order logic and a small knowledge base), it demonstrates that E is taking heed of his knowledge to generate a meaningful reply. In the RASCALS architecture, answering "Yes" to the query above implies that E must in fact have the corresponding knowledge, an implication that does not hold for other architectures.

For a more formal method of analysis, we introduce the "Knowledge Code Test": If synthetic character C says something X or does something X designed to evoke in the mind of the human gamer/user the belief that C knows $P_1, P_2, \ldots$, then we should find a list of formulas, or the equivalent, corresponding to $P_1, P_2, \ldots$ in the code itself. The characters in Figure 1 would fail such a test, as would characters built on the basis of Champandard's specifications. An FSA, as a matter of mathematical fact, has no storage capability. A system with power that matches that of a full Turing machine is needed to pass the Knowledge Code Test (Lewis & Papadimitriou 1981).

But formal proofs are oftentimes too detailed to be of interest. Before we can even begin translating a proof into an English justification, we need verify that its level of abstraction is high enough that it is easy to read and understand. After all, formal natural deduction proofs are difficult and tedious to read. To represent proofs at a more wholistic, abstract level, we utilize the denotational proof language known as Athena (Arkoudas 2000). Athena is a programming language, development environment, and interactive proof system that evaluates and processes proofs as input. Its most prominent feature is its ability to present proofs in an abstract, top-level manner, isomorphic to that of a natural argument a human might use. By developing proofs in Athena at this level, a level high enough to be of interest to a human reader, we can be sure that the language generated from our NLG module is at precisely the level of abstraction we desire — neither too detailed nor too amorphous.

It's now time to look at precisely how English is generated from a formal proof.

Proof-based Natural Language Generation

Very few researchers are experimenting with the rigorous translation of formal proofs into natural language[11]. This is particularly odd when one considers the benefits of such a program. Natural deduction proofs, provided that they are developed in a sensible manner, are already poised for efficient translation. They require absolutely *no* further document structuring or content determination. That is, document planning, as defined by (Reiter & Dale 2000), is completely taken care of by using formal proofs in the first place.

Our NLG module receives as input a formal proof and returns as output English text. The English generated is an isomorph of the proof received. The structure of the justification, then, is precisely the same as the structure of the proof. If the justification uses *reductio ad absurdum* in the middle of the exposition, then you can be sure that there's a proof by contradiction in the middle of the formal proof.

Formal proofs are constructed from various different subproofs. A proof by contradiction is one such example of a type of subproof, but there are of course many others. Our system breaks a proof down to its constituent subproofs, translating each subproof from the top down. For example, assume the following:

1. CHICAGO IS A TARGET OR NEW YORK IS A TARGET

[11]An example of one such team is a research group at the University of Saarlande. The group had, at least until 1997, been developing a system called PROVERB (Huang & Fiedler 1997). Their approach to proof-based translation was unique and extremely influential, though their project was largely unsuccessful.

2. If Chicago is a target, millions will die.
3. If New York is a target, millions will die.

To deduce something meaningful from this information, we'll use a proof by cases. Our system translates this proof form as follows:

Recall that Chicago or New York is a target. Each case produces the same conclusion; that is, if Chicago is a target then millions will die, and if New York is a target then millions will die. It follows that millions will die.

Predictably, documents produced in this manner, even when presented at a level abstract enough to make sense to a layperson, are rigid and, well, inhuman. They use the same phrases over and over again, they lack fluidity, and they are completely divorced of grace and wit. To boot, they disregard contextual information. Merely translating constituent subproofs to English will not produce natural English.

Nevertheless, this methodology provides a foundation for more sophisticated development. Once constituent subproofs are translated properly, they are sent to a microplanning system that maps particular subproofs to discourse relations (Hovy 1993). This mapping is known as a *message* and is not isomorphic. While the structure of the overall proof is preserved in the final document, individual subproofs are not treated with the same stringency. They can be molded and fitted to a number of different discourse relations for the sake of fluidity. Two more steps remain before natural language can be produced.

Lexicalization is the process by which a lexicon of words is selected and mapped onto its symbolic counterparts. The content implicit in the proof, structured through subproof analysis and discourse relations, needs to be lexicalized before it can be presented as English text. That is, exact words and phrases must be chosen to represent relationships and predicates. For instance, Target(Chicago) must be translated to Chicago is a target and Beretta(John) must be translated to John has a Beretta before we can move on to gluing everything together. The only way this can happen is if a lexical database such as WordNet (Miller 1995) is augmented with domain-specific lexicalizations such as those specifying how to lexicalize "Beretta AR-70."

For even more fluidity, it's necessary to avoid referring to the same entities with the same phraseology. At the very least, pronouns should be substituted when referring to repeated concepts, persons, places, and objects. These substitutions are known as *referring expressions*, and need to be generated to truly produce fluid, humanlike English.

Fortunately, once the above issues are resolved, the information gathered therein can be plugged easily into a surface realizer such as KPML (Bateman 1997). In this fashion, proof-based NLG allows for the generation of both structured and expressive expositions.

How far can an approach to NLG based on logic go? What about rhetorical structure, for example? The engineering of E reflects a belief that all of NLG, in the context of an advanced synthetic characters, can indeed be achieved through the mechanization of sufficiently complex logical systems. Only time will tell if this approach has the necessary breadth, but rhetorical structure seems particularly well-suited to capture in logic. Of note here is the fact that it was logic that dictated including the present paragraph.

E: The Presentation Level

To concretize our thoughts on evil, we show E; a realistic real-time presentation of an evil talking head in the formal sense. In order to give E a realistic look and a range facial expressions, we have created a muscle model of the face. Each simulated muscle in our model can contract and this contraction perturbs the vertices of the skin (E is rendered as a triangle mesh.) The effect of muscle simulation is supplemented by limited use of morph target based animation for some fine details. In addition, specialized actions are used to animate the eyes, jaw, and neck.

Prior Work

There have been several recent efforts in the presentation of talking heads. A VRML based approach shown by (Breton, Bouville, & Pel 2001) addresses all aspects of facial animation working in real-time, but for very low polygon models. The face is parameterized in a simplified manner similar to E.

A more complex, physics-based system is described in (Albrecht, Haber, & Seidel 2002). Here the focus is on lip synchronization and simulation of the mouth and lips. Other aspects of the face are not specifically addressed.

Our muscle simulation is based largely on that presented in (Waters 1987), (Parke & Waters 1996), and expanded upon in (Bui, Heylen, & Nijholt 2003). We chose to work from the Waters model because we feel it is most practical for real-time applications and implementation and programmable graphics hardware. We simulate two types of muscles.

Linear Muscles

Linear muscles contract along a single axis, and are parameterized by five values. The points O and T define the origin and terminus of the muscle respectively. The scalar F defines the radius from O where the effect of muscle contraction begins to decline. The angle Z defines the angle about O where the muscle affects the skin. Finally, the scalar W gives the distance between wrinkles that form as the muscle contracts.

We define a vertex shader (implemented in HLSL) that modifies the position of skin vertices based on muscle contractions. For a linear muscle this shader computes three values. Given a vertex at position P with normal N, the angular displacement, D_A, is:

$$\min(1 - \dot{(}norm(P-O), norm(T-O) * C, 0)$$

C is the muscle contraction in the above equation. The radial displacement, D_R, is:

$$1 - \left[\frac{clamp(len(P-O) - F, 0, len(T-O) - F)}{(len(T-O) - F)}\right]^3$$

Finally, the wrinkle offset, D_W, is:

$$\frac{1 - fmod(len(P-O), W) - (W * 0.5))^2}{(W * 0.5)^2}$$

We combine all these values to determine the final vertex displacement:

$$(C * D_A * D_R) + (D_W * N)$$

Many of the values in the above equations can be pre-computed before the vertex shader executes. The resulting implementation uses about 30 shader instructions.

Sphincter Muscles

Sphincter muscles draw together in circular shape and we use them to model the puckering lips and squinting eyes. A sphincter muscle is parameterized by an origin O, and a horizontal and vertical extent, H and V respectively. In the vertex shader the contraction of sphincter muscles displace vertices in the following manner:

$$\max \left[1 - \frac{\sqrt{((D_x^2 * V^2) + (D_y^2 * H^2))}}{H * V}, 0 \right] * C$$

In the above equation, D is the vector from a skin vertex to the origin of the muscle.

Putting It All Together

Each facial expression, be it a viseme used in speech or an emotional state, is described in terms of muscle contractions. To specify those contractions and drive E's facial animation systems, we use a very simplified scripting system for triggering named expressions at specified times with specified intensity values and blending parameters. We use the data from (Bui, Heylen, & Nijholt 2004) to prevent physically impossible muscle contractions. A parameterization for the tongue similiar to (King 2001) is used. A module for eye movements implements many of the ideas presented in (Lee, Badler, & Badler 2002). Finally, we simulate subsurface scattering on the skin using the algorithm of (Sander, Gosselin, & Mitchell 2004).

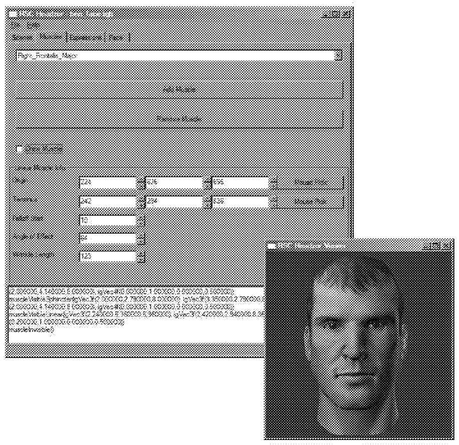


Figure 6: Tool for Manipulating Facial Muscles on E. (Note: Face shown *resembles* E's, but isn't his. E himself will be unveiled at the conference.)

Our Demos @ GameOn!

As mentioned above, at the conference we will show a conversation with E based on the first of the two aforementioned scenarios. This interaction will show our approach to the presentation level in action, and will manifest our formal account of evil in ordinary conversation that is based on our NLG technology.

References

[Albrecht, Haber, & Seidel 2002] Albrecht, I.; Haber, J.; and Seidel, H. P. 2002. Speech synchronization for physics-based facial animation. In *Proceedings WSCG*, 9–16.

[Anderson & Lebiere 1998] Anderson, J. R., and Lebiere, C. 1998. *The Atomic Components of Thought.* Mahwah, NJ: Lawrence Erlbaum.

[Anderson & Lebiere 2003] Anderson, J., and Lebiere, C. 2003. The newell test for a theory of cognition. *Behavioral and Brain Sciences* 26:587–640.

[Anderson 1993] Anderson, J. R. 1993. *Rules of Mind.* Hillsdale, NJ: Lawrence Erlbaum.

[Arkoudas & Bringsjord 2005] Arkoudas, K., and Bringsjord, S. 2005. Metareasoning for multi-agent epistemic logics. In *Fifth International Conference on Computational Logic In Multi-Agent Systems (CLIMA 2004)*, volume 3487 of *Lecture Notes in Artificial Intelligence (LNAI).* New York: Springer-Verlag. 111–125.

[Arkoudas 2000] Arkoudas, K. 2000. *Denotational Proof Languages.* Ph.D. Dissertation, MIT.

[Barwise & Etchemendy 1995] Barwise, J., and Etchemendy, J. 1995. Heterogeneous logic. In Glasgow, J.; Narayanan, N.; and Chandrasekaran, B., eds., *Diagrammatic Reasoning: Cognitive and Computational Perspectives.* Cambridge, MA: MIT Press. 211–234.

[Bateman 1997] Bateman, J. A. 1997. Enabling technology for multilingual natural language generation: the kpml development environment. *Nat. Lang. Eng.* 3(1):15–55.

[Breton, Bouville, & Pel 2001] Breton, G.; Bouville, C.; and Pel, D. 2001. Faceengine: a 3d facial animation engine for real time. In *Proceedings of 6th International Conference on 3D Web Technology*, 15–22.

[Bringsjord & Ferrucci 1998a] Bringsjord, S., and Ferrucci, D. 1998a. Logic and artificial intelligence: Divorced, still married, separated...? *Minds and Machines* 8:273–308.

[Bringsjord & Ferrucci 1998b] Bringsjord, S., and Ferrucci, D. 1998b. Reply to Thayse and Glymour on logic and artificial intelligence. *Minds and Machines* 8:313–315.

[Bringsjord & Ferrucci 2000] Bringsjord, S., and Ferrucci, D. 2000. *Artificial Intelligence and Literary Creativity: Inside the Mind of Brutus, a Storytelling Machine.* Mahwah, NJ: Lawrence Erlbaum.

[Bringsjord 1997] Bringsjord, S. 1997. *Abortion: A Dialogue.* Indianapolis, IN: Hackett.

[Bringsjord 1998] Bringsjord, S. 1998. Chess is too easy. *Technology Review* 101(2):23–28.

[Bringsjord 2001] Bringsjord, S. 2001. Is it possible to build dramatically compelling interactive digital entertainment (in the form, e.g., of computer games)? *Game Studies* 1(1). This is the inaugural issue. Url: `http://www.gamestudies.org`.

[Bringsjord forthcoming] Bringsjord, S. forthcoming. The RASCALS cognitive architecture: Logic top to bottom. In Sun, R., ed., *The Handbook of Computational Cognitive Modeling.* Cambridge University Press.

[Bui, Heylen, & Nijholt 2003] Bui, T. D.; Heylen, D.; and Nijholt, A. 2003. Improvements on a simple muscle-based 3d face for realistic facial expressions. In *Proceedings of 16th International Conference on Computer Animation and Social Agents*, 33–40.

[Bui, Heylen, & Nijholt 2004] Bui, T. D.; Heylen, D.; and Nijholt, A. 2004. Combination of facial movements on a 3d talking head. In *Proceedings of Computer Graphics International*.

[Cassell *et al.* 1999] Cassell, J.; Bickmore, T.; Billinghurst, M.; Campbell, L.; Chang, K.; Vilhjalmsson, H.; and Yan, H. 1999. Embodiment in conversational interfaces: Rea. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, 520–527. New York, NY, USA: ACM Press.

[Cassimatis *et al.* 2004] Cassimatis, N.; Trafton, J.; Schultz, A.; and Bugajska, M. 2004. Integrating cognition, perception and action through mental simulation in robots. In *Proceedings of the 2004 AAAI Spring Symposium on Knowledge Representation and Ontology for Autonomous Systems*.

[Cassimatis 2002] Cassimatis, N. 2002. *Polyscheme: A Cognitive Architecture for Integrating Multiple Representation and Inference Schemes*. Ph.D. Dissertation, Massachusetts Institute of Technology (MIT).

[Champandard 2003] Champandard, A. 2003. *AI Game Development*. Berkeley, CA: New Riders.

[Electronic Arts Inc. 2000] Electronic Arts Inc. 2000. *The Sims^{TM}: The People Simulator from the Creator of SimCity^{TM}*. Austin, TX: Aspyr Media.

[Feinberg 2003] Feinberg, J. 2003. *Problems at the Roots of Law*. New York, NY: Oxford University Press.

[Genesereth & Nilsson 1987] Genesereth, M., and Nilsson, N. 1987. *Logical Foundations of Artificial Intelligence*. Los Altos, CA: Morgan Kaufmann.

[Goldstein 2005] Goldstein, E. B. 2005. *Cognitive Psychology: Connecting Mind, Research, and Everyday Experience*. Belmont, CA: Wadsworth.

[Gratch *et al.* 2002] Gratch, J.; Rickel, J.; Andre, E.; Cassell, J.; Petajan, E.; and Badler, N. 2002. Creating interactive virtual humans: Some assembly required. *IEEE Intelligent Systems* 17(4):54–63.

[Horty 2001] Horty, J. 2001. *Agency and Deontic Logic*. New York, NY: Oxford University Press.

[Hovy 1993] Hovy, E. H. 1993. Automated discourse generation using discourse structure relations. *Artif. Intell.* 63(1-2):341–385.

[Huang & Fiedler 1997] Huang, X., and Fiedler, A. 1997. Proof verbalization as an application of NLG. In Pollack, M. E., ed., *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, 965–970. Nagoya, Japan: Morgan Kaufmann.

[King 2001] King, S. A. 2001. *A Facial Model and Animation Techniques for Animated Speech*. Ph.D. Dissertation, Ohio State University.

[Lee, Badler, & Badler 2002] Lee, S. P.; Badler, J. B.; and Badler, N. I. 2002. Eyes alive. *ACM Transactions on Graphics* 21(3):637–644.

[Lewis & Papadimitriou 1981] Lewis, H., and Papadimitriou, C. 1981. *Elements of the Theory of Computation*. Englewood Cliffs, NJ: Prentice Hall.

[Miller 1995] Miller, G. A. 1995. Wordnet: a lexical database for english. *Commun. ACM* 38(11):39–41.

[Newell 1973] Newell, A. 1973. You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In Chase, W., ed., *Visual Information Processing*. New York: Academic Press. 283–308.

[Nilsson 1991] Nilsson, N. 1991. Logic and Artificial Intelligence. *Artificial Intelligence* 47:31–56.

[Parke & Waters 1996] Parke, F. I., and Waters, K. 1996. *Computer Facial Animation*. New York, NY: AK Peters.

[Peck 1983] Peck, M. S. 1983. *People of the Lie*. New York, NY: Simon and Shuster.

[Pelachaud & Poggi 2002] Pelachaud, C., and Poggi, I. 2002. Multimodal embodied agents. *Knowl. Eng. Rev.* 17(2):181–196.

[Reiter & Dale 2000] Reiter, E., and Dale, R. 2000. *Building natural language generation systems*. New York, NY, USA: Cambridge University Press.

[Rinella, Bringsjord, & Yang 2001] Rinella, K.; Bringsjord, S.; and Yang, Y. 2001. Efficacious logic instruction: People are not irremediably poor deductive reasoners. In Moore, J. D., and Stenning, K., eds., *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*. Mahwah, NJ: Lawrence Erlbaum Associates. 851–856.

[Ritter *et al.* June 2002] Ritter, F.; Shadbolt, N.; Elliman, D.; Young, R.; Gobet, F.; and Baxter, G. June 2002. Techniques for modeling human performance in synthetic environments: A supplementary review. Technical report, Human Systems Information Analysis Center, Wright-Patterson Air Force Base, OH.

[Rosenbloom, Laird, & Newell 1993] Rosenbloom, P.; Laird, J.; and Newell, A., eds. 1993. *The Soar Papers: Research on Integrated Intelligence*. Cambridge, MA: MIT Press.

[Russell & Norvig 2002] Russell, S., and Norvig, P. 2002. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall.

[Sander, Gosselin, & Mitchell 2004] Sander, P. V.; Gosselin, D.; and Mitchell, J. L. 2004. Real-time skin rendering on graphics hardware. In *Proceedings of ACM SIGGRAPH*.

[Sun 2001] Sun, R. 2001. *Duality of the Mind*. Mahwah, NJ: Lawrence Erlbaum Associates.

[Waters 1987] Waters, K. 1987. A muscle model for animating three-dimensional facial expression. In *Proceedings of ACM SIGGRAPH*, volume 21, 17–24.

[Yang & Bringsjord 2001a] Yang, Y., and Bringsjord, S. 2001a. Mental metalogic: A new paradigm for psychology of reasoning. In *Proceedings of the Third International Conference on Cognitive Science (ICCS 2001)*. Hefei, China: Press of the University of Science and Technology of China. 199–204.

[Yang & Bringsjord 2001b] Yang, Y., and Bringsjord, S. 2001b. The mental possible worlds mechanism: A new method for analyzing logical reasoning problems on the gre. In *Proceedings of the Third International Conference on Cognitive Science (ICCS 2001)*. Hefei, China: Press of the University of Science and Technology of China. 205–210.

[Yang & Bringsjord 2005] Yang, Y., and Bringsjord, S. 2005. *Mental Metalogic: A New, Unifying Theory of Human and Machine Reasoning*. Mahway, NJ: Erlbaum.

[Yang & Johnson-Laird 2000a] Yang, Y., and Johnson-Laird, P. N. 2000a. How to eliminate illusions in quantified reasoning. *Memory and Cognition* 28(6):1050–1059.

[Yang & Johnson-Laird 2000b] Yang, Y., and Johnson-Laird, P. N. 2000b. Illusory inferences with quantified assertions. *Memory and Cognition* 28(3):452–465.

[Yang, Braine, & O'Brien 1998] Yang, Y.; Braine, M.; and O'Brien, D. 1998. Some empirical justification of one predicate-logic model. In Braine, M., and O'Brien, D., eds., *Mental Logic*. Mahwah, NJ: Lawrence Erlbaum Associates. 333–365.

Simulation and Modelling of Adversarial Games

Erol Gelenbe, Varol Kaptan and Yu Wang
Department of Electrical and Electronic Engineering
Imperial College London
Exhibition Road, London SW7 2AZ
e-mail: {e.gelenbe|v.kaptan|yu.wang3}@imperial.ac.uk

KEYWORDS
AI, Simulation & Modelling, Autonomous Agents, Navigation.

ABSTRACT

Simulation is an important tool for both training and assessment of systems of adversarial agents involved in a conflict, especially in the case of military tactical scenarios where real exercises tend to be very costly and come with a high risk of injury for the participants. One of the challenges of tactical simulations is the ability to represent a large number of agents, which while acting as individuals are also embedded in the social structure of the groups involved. In this paper, we present a behaviour-based model of agent control which can model aspects of both individual and group behaviour. Within this context, we describe our approach to solving two questions of current interest. The first one is how to obtain, through mathematical modelling, estimates of long-term and average behaviour of an agent system in a timely manner. The second is how to adapt agent models to take advantage of specific features of the urban environment, which appears to be the primary theatre of operations of modern conflicts.

INTRODUCTION

Discrete event simulation is widely used to model, evaluate and explore operational contexts of real systems under varying synthetic conditions. Simulation runs can predict the capabilities and limitations of different operational rules or of different combinations of tactical assets. An important aspect of agent simulations, within this context, is the realism of agent behaviour. Depending on the particular application domain, agents which exhibit very limited or even very advanced intelligence may be considered unrealistic. This is especially true in the context of simulations designed for training personnel or evaluating tactical situations. Unfortunately, human behaviour being a product of natural intelligence is a very complex topic and is still not well understood.

Traditionally, discrete event simulation has concentrated on the algorithmic description and control of synthetic entities which are being modelled as they accomplish some meaningful function, and simulation research has devoted much attention to appropriate workload representation and output data analysis.

Less attention has been paid to the design of simulation systems in which individual animated objects (such as manned or robotic vehicles, or human individuals) are provided with broad goals (such as "go quickly to that hill, and do not get killed") and are then allowed to dynamically attain their objective through emergent behaviour based on individual adaptation and learning (Gelenbe 1999; Gelenbe et al. 2000; Gelenbe et al. 2001). Our research has been inspired by the need to be able to simulate a large number of concurrent agents through methods which offer a balanced trade-off between realism and computational efficiency.

The rest of the paper consists of three main sections. In the first section, we consider how a variety of adaptive paradigms, including reinforcement learning, social potential fields and imitation, can be used in a simulation to investigate how the simulated entities may attain broadly defined goals without detailed step-by-step instructions within a physically precise environment. In the second section, we describe a novel approach for modelling large-scale agent systems with similar capabilities through stochastic population models. The last section describes a method for adapting our simulation and modelling algorithms to complex terrains with particular emphasis on urban environments.

SIMULATING COLLECTIVE AUTONOMOUS BEHAVIOUR

To illustrate our approach to modelling collective autonomous behaviour we will focus on the problem of goal-based navigation of a group of autonomous entities in a dangerous terrain. The design of our agent model is based on the assumption that agents will perform *outdoor* missions in a terrain containing simple obstacles and enemies. In the last section we will show how this model can be adapted to more specialised terrains like urban environments.

A *mission* in our model is defined as the problem of going from some position A to some other position B avoiding being hit by an enemy, and avoiding the natural and artificial obstacles present in the terrain. The success of the mission can be measured by the amount of time necessary for the whole group to achieve the goal and its the survival rate, for example.

Different decision mechanisms are used to model different aspects of the agent behaviour and a higher level coordination module is combining their output. Such an architecture allows *versatile agent personalities* both in terms of hetero-

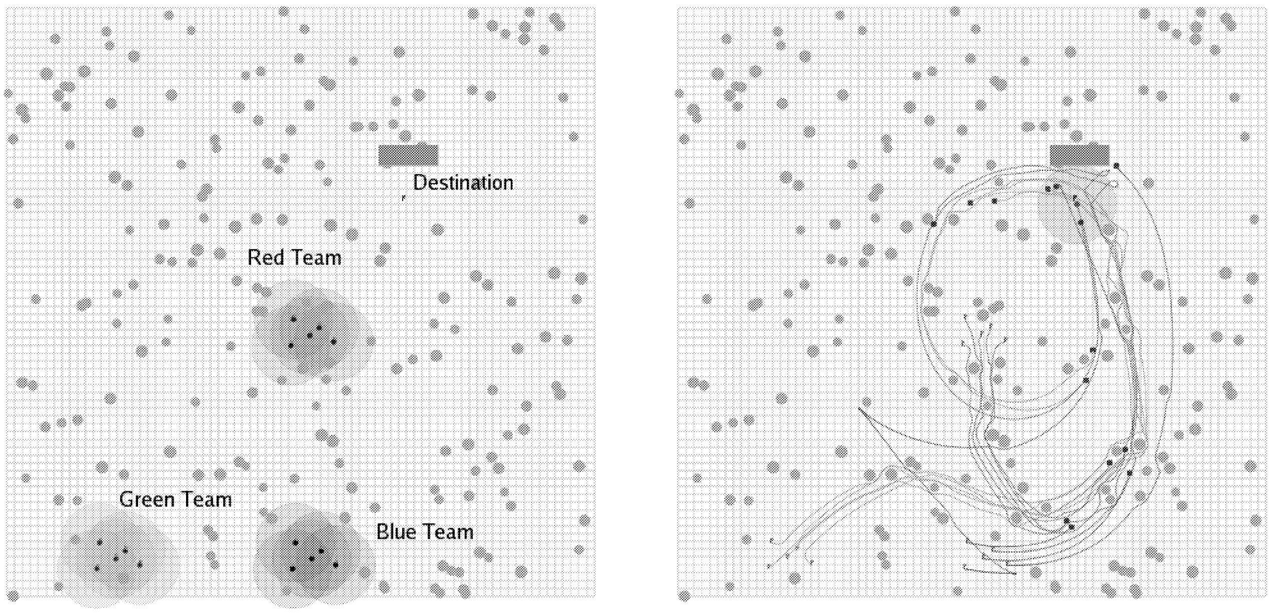


Figure 1: Screenshots of the agent simulator: (a) left, scenario configuration; (b) right, an example mission outcome

geneity (agent specialisation) within a group and dynamic (i.e. mission-context sensitive) agent behaviour.

In our current model, we have three basic modules that we call the *navigation* module, *grouping* module and *imitation* module.

- The Navigation Module is responsible for leading a single agent from a starting location to a destination location, avoiding danger and obstacles.
- The Grouping Module is responsible for keeping a group of agents together in particular formations throughout the mission.
- The Imitation Module is modelling the case when an inexperienced agent will try to mimic the behaviour of the most successful agents in the group and thus increase its chances of success.

In the most simple case, the navigation behaviour can be modelled simply as moving in such a way as to minimise the time it will take to go to the destination. When travelling in a dangerous environment, an agent can also take into account the probability of being incapacitated/destroyed when estimating the transition time to destination based on terrain information provided *a priori* or acquired through local observations.

Group behaviour is modelled through Social Potential Fields (SPFs) (Reif and Wang 1995). In order to "encourage" members of an agent team to keep a certain spatial configuration while performing a mission, we set a combination of attractive and repulsive forces between the respective team members. The parameters of the forces can be adjusted to generate a force profile which is attractive at long distances and repulsive at short distances. By varying both force parameters and force connectivity, a wide range of spatial formations with different sizes can be achieved.

Some of these approaches may incorporate memory (navigation) while others others can be purely reactive (grouping) and some may depend on the performance of other members in an agent group (imitation and grouping).

Figure 1(a) describes a particular scenario involving three groups of agents colour-coded as the Red, Green and Blue teams. The Blue team has the goal to go to the destination and avoid conflicts with adversaries in the process. The Red team has the goal to engage the Blue team. The goal of the Green team is to help the Blue team by engaging the Red team. Figure 1(b) shows a particular outcome of this mission. The exact details of the agent control algorithms are described in (Gelenbe et al. 2004).

MATHEMATICAL MODELS OF AGENT SIMULATION

The world that we live in is filled with large scale agent systems, from diverse fields such as biology, ecology or finance. Inspired by the desire to better understand and make the best out of these systems, we are extending the detailed simulation work that we describe in the preceding paragraphs, to an approach based on building stochastic mathematical models, in particular G-networks (Gelenbe 1994; Fourneau et al. 1996) models from the simulation. We aim to provide insight into systems in terms of their performance and behaviour, to identify the parameters which strongly influ-

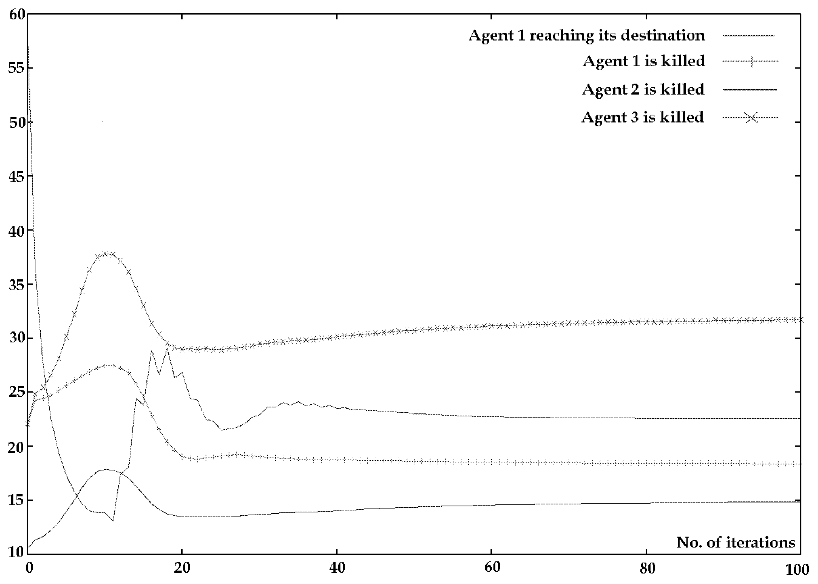


Figure 2: Example results of a stochastic agent model: expected times for certain events of interest

ence them, and to evaluate how well individual goals can be achieved.

Quite often a metric of interest will happen to be defined over an ensemble of simulation runs. In order to get meaningful estimates of mission outcome one has to collect data over many simulations to get statistically significant results. We propose an approach where some of this information can be provided by a mathematical framework based on stochastic population models. Such models can be used by themselves as well as to complement results obtained through discrete event simulation of adversarial tactical scenarios.

We model the agents and their interactions as a G-network where each agent class has a queue at each location and agents travelling and interacting in the environment are modelled as customers being serviced by their respective queues. The G-network model is quite capable of representing the spectrum of interactions available to the agent model described in the previous section, like goal-based navigation, group behaviour, adversarial action, etc. It can even represent situations like changing plans (e.g. destination or target), splitting forces to carry out alternative plans, or even agents mutating into a different class (agents switching sides, for example).

We can assess the performance of an individual or group goal, the time it takes for a group to achieve its goal, the survival rate of a group, etc., by using the equilibrium probability distribution of an agent's location which can be found from the steady state-solution of the respective G-network. Figure 2 shows an example result of a scenario similar to the one described in the previous section (the terrain is quantised to a 15×15 grid, for simplicity of presentation, each group has only one agent). The graph shows time estimates of certain events (i.e. expected time to reach a destination, average live expectancies of the different agents) as a function of the number of fixed-point iterations over the G-network. In this example it takes only 40-50 iterations for the model to converge to steady state.

This mathematical model can also be used to model systems at different abstraction levels, in terms of the number of agents or the size of the geographical location. This allows us to extrapolate results for smaller systems as a means of achieving faster convergence when solving a system at a higher level of detail. In doing so, we can greatly reduce computational complexity and save time and resources.

SIMULATING AND MODELLING AGENTS IN URBAN ENVIRONMENTS

One of the main problems of predominantly-reactive behaviour control techniques like the methods of simulating group behaviour described in the first section is that these methods suffer from the local minima problem. This problem is common in mathematics when searching for parameters which globally maximise or minimise the value of a function - simple search approaches can be easily trapped at a local extremum point. In the problem of agent navigation, getting stuck in a local minima can be due to either the particular agent configuration or extra constraints imposed on the agents by the terrain. The former is difficult to quantify, since the force profiles involved in the agent model usually encode only the rough intent of a mission designer and maybe based on intuition. As such it is quite subjective since it can be argued that a particular outcome was intended or can be considered good enough for a particular purpose.

In this section we will focus on the latter which is usually more troublesome, especially in urban environments. Let us illustrate the problem with a very simple example - suppose an agent is embedded in an environment where its motion is controlled through social potential fields or similar methods. The terrain includes some obstacles which have to be avoided. In the case when the collision avoidance scheme adversely affects the motion towards an intended goal, the agent may either become stationary or be stuck on a localised

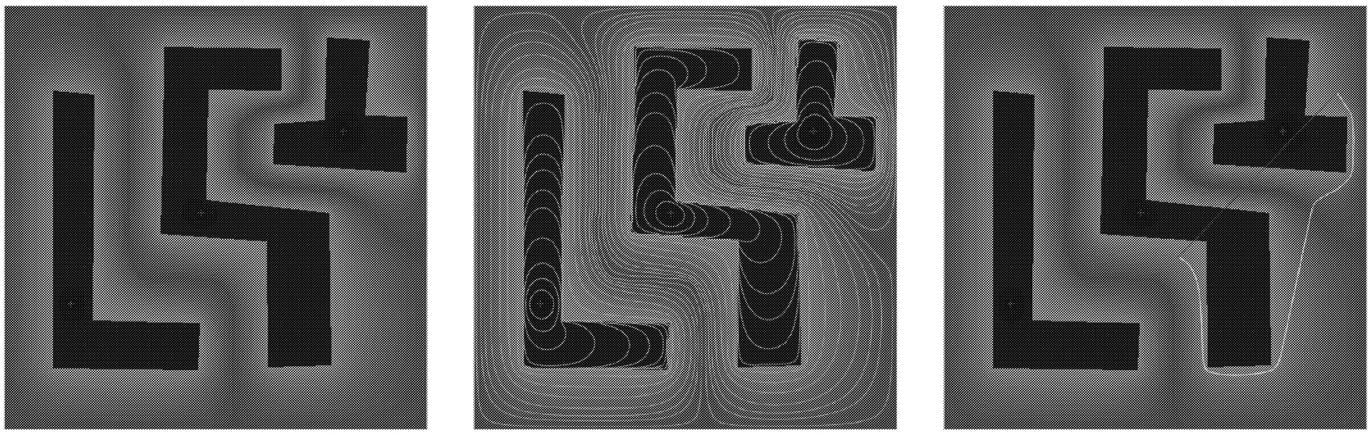

(a) Some buildings (b) Obstacle implosion (c) An example of a clear path

Figure 3: Example of urban terrain navigation

cyclic path which does not get it nearer to the final goal.

Our approach to dealing with obstacles is based on the idea of finding a transformation between the real navigation space and a virtual obstacle-free space and applying the classical control methods within this new space. A complete mathematical description of our methods is beyond the scope of this paper so we will try to illustrate our approach in a more informal but accessible as follows.

1. We assume that all obstacles have a continuous boundary which is a closed contour. The obstacles occupy the inside of the contours. The space outside the contours is free (i.e. accessible to agents).
2. We continuously shrink the contour inwards until the obstacles collapse into a point singularity. The contour will stretch the surrounding free space in the process of shrinking.
3. When all the obstacles have collapsed, the free space would span the whole real terrain and any point in the morphed free space will be accessible from any other point trivially, by navigating along a straight line.
4. It is very important that our transform function conserves the local continuity of the space in the process - this property guarantees that a straight line in the morphed space is a continuous curve (thus a valid navigation path) in the real space.

The above transform can be used for controlling groups of agents in urban environments in the following way: When a goal or an object exerting force on an agent is within a line of sight, the agent can base its behaviour on application of the respective control methods within the real space. When this is not the case the agent will simply "switch" to operating in the virtual space where an unobstructed line of sight, while not necessarily optimal, is at least guaranteed. The approach is graphically illustrated in Figure 3. The picture on the left shows a birds-eye view over a simple terrain containing three non-convex obstacles (buildings). The picture in the middle shows how the shape of the obstacles gradually collapse into singularities during the transformation of the terrain. The picture on the right shows two locations on the terrain with no clear line of sight (connected with a straight line) and the respective route that is generated while navigating with the help of the transformation to the virtual obstacle-free space.

One of the main advantages of such an approach is that it can be used to provide improvement to heuristic or reactive control methods as used in a wide range of application domains where due to computational or time constraints classical path planning and optimisation methods are not feasible.

CONCLUSIONS AND FUTURE WORK

In this paper, we presented a behaviour-based approach for modelling groups of adversarial agents. In particular, we described some recent results on building G-networks inspired mathematical models which allow the estimation of long-term and steady state properties of such agent systems without having to rely on gathering statistics from lengthy simulation runs. The other contribution of this paper is a novel method of transforming an urban environment with many complex obstacles into a virtual obstacle-free space which allows the application of a number of mostly reactive (but usually very efficient and scalable) techniques to the domain of adversarial action in an urban environment.

Possible future directions for research include qualitative comparison of results obtained through simulation versus modelling, especially with respect to the spatial resolution at which a G-network models a simulated scenario. We are also planning to perform a quantitative analysis of the performance of our urban terrain approach with respect to a classical path-planning approach, both in term of computational cost and optimality.

REFERENCES

Fourneau, J.-M.; Gelenbe, E.; and Suros, R. 1996. “G-networks with multiple classes of positive and negative customers”. *Theoretical Computer Science*, 155:141–156.

Gelenbe, E. 1994. “G-networks: An unifying model for queuing networks and neural networks”. *Annals of Operations Research*, 48(1-4):433–461.

Gelenbe, E. 1999. “Modeling CGF with learning stochastic finite-state machines”. In *Eighth Conference on Computer Generated Forces and Behavioral Representation*, pages 113–115, Orlando, Florida.

Gelenbe, E.; Kaptan, V.; and Hussain, K. 2004. “Simulating the navigation and control of autonomous agents”. In Svensson, P. and Schubert, J., editors, *Proceedings of the Seventh International Conference on Information Fusion*, volume I, pages 183–189, Mountain View, CA. International Society of Information Fusion.

Gelenbe, E.; Şeref, E.; and Xu, Z. 2000. “Discrete event simulation using goal oriented learning agents”. In *AI, Simulation & Planning in High Autonomy Systems*, Tucson, Arizona. SCS.

Gelenbe, E.; Şeref, E.; and Xu, Z. 2001. “Simulation with learning agents”. *Proceedings of IEEE*, 89(2):148–157.

Reif, J. H. and Wang, H. 1995. “Social potential fields: A distributed behavioral control for autonomous robots”. In Peters, A. K., editor, *International Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 431–459, Wellesley, Massachusetts.

GAME PHYSICS AND FACIAL ANIMATION

ENHANCING GAME PHYSICS USING GAUSS MAP COMPUTATION

L. Alboul, G. Echeverria,* and M. Rodrigues
Geometric Modelling and Pattern Recognition Group
Materials and Engineering Research Institute
Sheffield Hallam University
City Campus, S1 1WB, Sheffield, United Kingdom
e-mail: l.alboul@shu.ac.uk, g.echeverria@shu.ac.uk, m.rodrigues@shu.ac.uk

KEYWORDS
Gauss Map, Discrete Curvature, Polyhedral Surfaces, Game Physics, Game Terrain

ABSTRACT

The present paper presents an algorithm to make fast and simple computations of the surface curvature of polyhedral objects in 3D, using a method called the Polyhedral Gauss Map. The computations are simple and can be done either in advance for static objects that are not modified during the game, or in real time, for dynamically changing entities. The curvature information can be applied for the computations of the physical properties and behaviour of objects in a game, for example, estimating the 'roughness' of terrain or measuring the curvature of a race track in order to find the optimal path or identify objects of different curvature when sliding down a sloped surface.

INTRODUCTION: GAME PHYSICS

Modern computer and console games look to represent a more believable simulation of environments. Recent advances in technology mean more powerful platforms with much more resources available to developers. However, as the technological possibilities increase, so does the required amount of realism expected from games. The area with the most improvement has generally been graphics. But graphics alone do not completely convey reality. The objects in the game must also behave in a natural way. With better graphics, it is expected a greater level of credibility in all other aspects of the game, such as the interaction of the in-game objects.

In order to represent an accurate motion of game objects, a physical model is used, which will compute the appropriate behaviour of each object. Unfortunately game systems have limited processing resources, which must be divided among several tasks: processing of graphics, game logic, artificial intelligence, access to peripherals, as well as the computation of the physics model. For this reason, it is necessary to use simple and effective algorithms to compute physical properties of objects, without using too much processing time.

Having extra information about the properties of an object can improve the simulation of its behaviour and interaction with other similar objects. Measurements of shape parameters of an object, for example, of how 'rounded' an object is, can help to convey more believable reactions. This paper presents a new method, based on curvature computation and visualisation, that allows an efficient shape description of complex objects, especially surfaces. The method might be useful in various games genres, in particular in action-adventure games, racing games, sport and puzzle games, and educational games [Quize 2003].

CURVATURE

Among physical properties of an object one of the most important is the object's shape. In computer games objects used are mostly of two types: two-dimensional surfaces and three-dimensional solid objects. A solid object is also commonly represented by its boundary, which is, in general, a surface. In games, objects are often undergoing transformation and deformation. Therefore, a thorough and quick apprehension of a surface shape at various moments of the game is very important.

The curvature of an object is a good measure of how it's shape varies from one point to another in its surface. Two main curvatures: *Gaussian curvature* and *Mean curvature* are basic measures to describe local shape of a smooth surface. Roughly speaking, the Gaussian curvature of the surface is the measure of the 'deviation' of the surface from a plane and the Mean curvature is the measure of its 'bending' in space.

For smooth objects, curvature can be computed at each point using methods of calculus. However, most of the surfaces of game objects are not smooth and represented by triangular or polygonal meshes, which are common in computer-related applications. In these cases, a different approach to determine curvature is required which will be analogous to the method for smooth surfaces.

If we consider integral relations for curvature of smooth surfaces, then the basic concept behind curvature measures, namely the concept of the *angle* appears explicitly. This can be illustrated by means of the Gauss Map.

*Corresponding author

Integral curvature measures are to be taken then as the main quantities to determine curvature analogues for polygonal meshes.

GAUSS MAP

For a domain U of smooth surface S the Gauss map $N(U)$ is the map assigning to each point $p \in U$ the point on the unit 2-sphere $S^2 \in \mathbf{R}^3$, by 'translating' the unit normal vector $N(p)$ to the origin [Kühnel 2002]. If $U(p)$ is small enough that the map $N(U(p))$ is one-to-one then normals will form a solid angle, whereas the end-points of normals will carve a certain region on S^2 (see Figure 1). The area of this region, numerically equal to the measure of the solid angle, formed by normals, is the measure of (integral) curvature of the region $U(p)$. This measure is considered positive if the map $N(U(p))$ is orientation-preserving (outward normals at corresponding points correspond), and negative otherwise. In the former case $U(p)$ represents is a convex (concave) region, and in the latter case it is a saddle region.

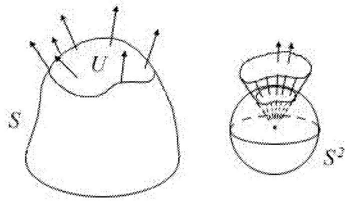


Figure 1: Gauss Map of a smooth surface

The (classical) Gaussian curvature $K(p)$ at a point p is then defined by setting $|K(p)| = \lim_{U(p)\downarrow p} \frac{AreaN(U(p))}{AreaU(p)}$ where the limit is taken as the neighbourhood $U(p)$ contracts down to the point p.

If the map $N(U(p))$ is not one-to-one then the region $U(p)$ can be represented as the union of non-overlapping subregions $U_\alpha(p)$ for each of which $N(U_\alpha(p))$ is one-to-one. Different subregions can be mapped to the same region on the unit sphere, which results in multiplicities of the Gauss map.

Under the integral Gaussian curvature one understands the algebraic area of the image $U(p)$ under the Gauss mapping:

$$K_{int} = \int_U KdU. \tag{1}$$

If we define on S the function $K^+ = \max K, 0$ (i.e. equal to the Gaussian curvature K if K is positive, and zero wherever K is zero or negative) then for the region $U(p)$ the positive integral curvature defined as $K_{pos} = \int_U K^+ dU$. The integral $\int_S |K| dU$ is called the total absolute curvature of U.

DISCRETE CURVATURE

Designating $\mathbf{V}$ as a finite point set in three-dimensional space, $\mathbf{V} = \{V_i, i = 1, \ldots, n\}$; then by $\mathbf{P}_k(\mathbf{V}, \mathbf{E})$ we denote a polyhedral surface (aka discrete surface) with the vertex set $\mathbf{V}$ and edge set $\mathbf{E}$. It represents a union of finite number of planar polygonal region, called *facets.* Without loss of generality we assume that each polygonal region is a triangle, referred further to as a *face.*

Curvature of a polyhedral surface is expressed in terms of the angles created around its vertices (analogue to the integral Gauss curvature) and along the edges (analogue to the the integral Mean curvature). A vertex ν together with the faces to which it belongs forms the *star* $Str(\nu)$ of the vertex.

The method proposed here measures both types of curvatures but we mostly concentrate on the curvature around the vertices.

Currently existing methods to analyse curvature rely only on measuring the angles of the faces around a vertex (see, for example, [Yamauchi et al. 2005]). However, this method is not sufficient, as it only obtains the correct result for convex vertices or pure saddles, but fails whenever a vertex is of another type.

Discrete Curvatures and Types of Vertices

In the previous section several integral curvature measures were listed: the integral Gaussian curvature, integral positive curvature, and total absolute curvatures. For discrete surfaces each of these measures has an analogue, although not always a straightforward one. Such extensions are described as follows.

Curvature ω around vertex ν: The curvature ω around the vertex, which is analogous to the integral Gaussian curvature, and therefore often simply referred to as the *Gaussian curvature*, is defined as follows:

$$\omega = 2\pi - \theta \tag{2}$$

where $\theta = \sum \alpha_i$ is the total angle around vertex ν, and α_i are those angles of the faces in the $Str(\nu)$ that are incident to ν (see Figure 2).

The value returned for ω is also called the *angle deficit* of the vertex, and it can be computed for any point x in $\mathbf{P}$, but only for vertices ω might be not equal to zero.

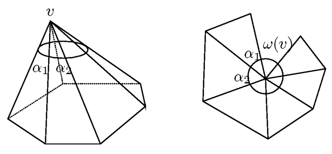


Figure 2: Computation of the Gaussian Curvature (ω) around vertex ν

For a domain $U \subseteq \mathbf{P}$ then we have

$$\Omega_U = \sum_{\nu \in U} \omega_\nu. \qquad (3)$$

Let us note that the positive value of $\omega(\nu)$ does not guarantee that the vertex is convex, as well as the negative value does not guarantee that ν represents a saddle, as we show below.

Positive (extrinsic) curvature ω^+: The following measure to be determined is an analogue of the (integral) positive curvature for a polyhedral domain. However, if we look at the picture in Figure 3, we can see that in both polyhedra all curvatures ω_ν are positive and actually are equal for every corresponding vertex.

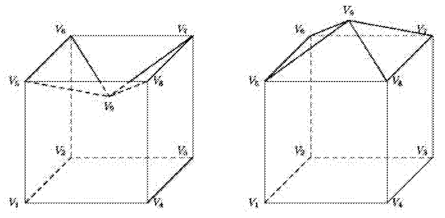


Figure 3: Two polyhedra with equal Gaussian curvature

Therefore, we have:

$$\Omega(P_1) = \Omega(P_2) = \sum_{\nu \in P_1} |\omega_\nu| = \sum_{\nu \in P_2} |\omega_\nu| = 4\pi. \qquad (4)$$

But for the smooth compact surface S the following inequality takes place: $\int_S K^+ dA \geq 4\pi$, which is equal to 4π only for convex surfaces. The left polyhedron is nonconvex, and the above equation does not reflect this fact. The problem is that positive and negative 'parts' of the curvature, if they exist, are 'glued' together; but it is not difficult to separate them. If vertex ν belongs to the boundary of the convex hull of its star (i.e. the convex hull of ν and all vertices in its star), then we can single out another star $\mathbf{Str}^+(\nu)$ with ν as the vertex and those edges of $\mathbf{Str}(\nu)$ that belong to the boundary of the convex hull. The edges of $\mathbf{Str}^+(\nu)$ will determine the faces of $\mathbf{Str}^+(\nu)$. We refer to $\mathbf{Str}^+(\nu)$ as the convex cone of vertex ν. Then

$$\omega^+ = 2\pi - \theta^+ \qquad (5)$$

where θ^+ is the total angle around ν in $\mathbf{Str}^+(\nu)$. If the convex cone around ν doesn't exist, i.e. ν lies inside the convex hull of $\mathbf{Str}(\nu)$, then ω^+ is, by definition, equal to zero. This would be the case of a saddle vertex.

Negative (extrinsic) curvature ω^-: We can now 'extract' the negative part of ω as follows:

$$\omega^- = \omega^+ - \omega \qquad (6)$$

This component will be different from zero in vertices which have at least one concavity, or on saddles.

Absolute (extrinsic) curvature ω_{abs}:

$$\omega_{abs} = \omega^+ + \omega^- \qquad (7)$$

The curvature of an entire object is computed as the sum of the Absolute Curvatures for each one of its vertices, and is referred to as the *Total Absolute Curvature.*

Considering these types of curvatures, vertices of a polyhedral object can be classified as: *Convex vertices* ($\omega^+ = \omega$), *Saddle vertices* ($\omega^- = -\omega$) and *Mixed vertices* ($\omega^+ > 0, \omega^+ \neq \omega$). Examples of all these vertices are shown in Figure 4.

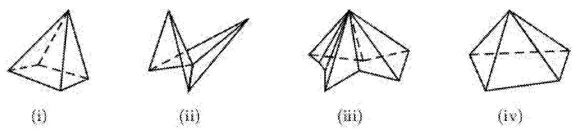


Figure 4: Examples of vertices: convex (i), saddle (ii), and mixed (iii) with its convex cone (iv)

In computer-aided applications the *angle deficit* computation is commonly used [Meyer et al. 2003, Peng et al. 2003] , but other types of curvature seem to be almost completely unrecognised, which results in losing important features of objects under consideration.

POLYHEDRAL GAUSS MAP

The computation of curvature of a discrete surface as defined above can be represented graphically in a sphere, by analogy with the Gauss map for smooth surfaces.

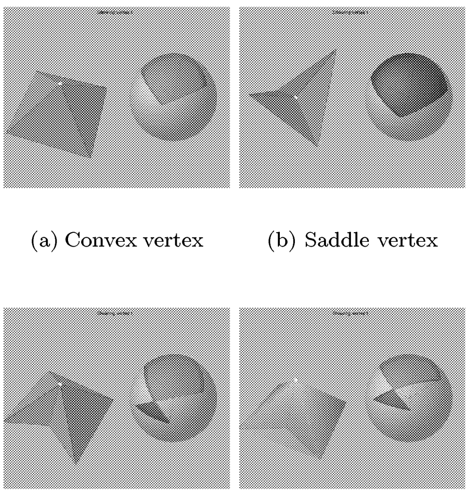
To compute the Gauss Map of a single vertex ν, the procedure is to take the normal vectors of all faces in $Str(\nu)$. All these vectors are translated so that they have the same origin, and their end-points lie on the surface of a sphere.

The next step is to join the ends of the vectors, ordered according to the corresponding faces around the vertex, in counter clockwise direction. The vectors are linked using arc segments of great circles on the sphere.

One or more areas will be delimited on the surface of the sphere by the arcs, thus creating a series of spherical polygons, each one of which represents a certain curvature feature of the vertex. Positive curvature will generate spherical polygons with counter clockwise orientation, while a negative curvature creates a spherical polygon with clockwise orientation.

However, each of these spherical polygons will split the sphere into two areas, complement of each other. The algorithm has to determine which of these areas is the correct one. A number of parameters are used to determine this, depending on the characteristics of the vertex. For vertices without self-intersections, all areas should be less than 2π, but otherwise, other checks are carried out: whether the vertex is a saddle or not, and the length of the arcs delimiting the area (the spherical perimeter) being less than 2π.

The correct values for each curvature are then directly computed from adding the area of the spherical polygons obtained using this method. The same procedure is then carried on for all required vertices, to obtain the Total Absolute Curvature (TAC), which will be the sum of the areas of all spherical polygons, taking into consideration their orientation. Figure 5 shows the Gauss maps of the basic types of vertices, described in the previous section (red colour areas correspond to the incorporated positive curvature, and the blue colour, to the negative one). However, this algorithm is capable of determining curvature measures of much more complex vertices with various self-intersections (for more detail see [Alboul and Echeverria 2005]).



(a) Convex vertex (b) Saddle vertex

(c) Mixed vertex (d) Mixed (with convex hull)

Figure 5: Gauss Map of different kinds of vertices

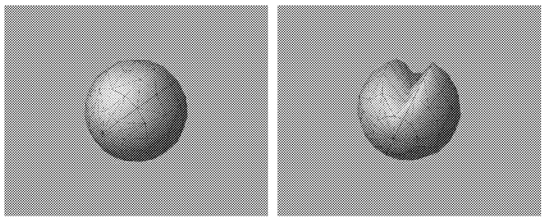
APPLICATIONS OF THE CURVATURE INFORMATION

In a video game, curvature can be computed for individual objects, to determine whether they have round or sharp vertices, and thus, if the object would present more resistance or friction. The information provided by the curvature analysis can be as simple as a single number representing the Total Absolute Curvature of the object, or a series of numbers each characterising a specific part.

The analysis of curvature proposed is effective even in the case of self intersecting polygonal meshes, making it useful for almost any kind of object in a video game.

This kind of application can be also useful in puzzle or sport games. For example, the curvature of a ball could be used to simulate different behaviours, or the wear of a ball during the game.

In the most general case, any object that is completely convex and has no 'indents' or saddle-type parts, such as a sphere or a cube, will have a TAC = 4π, while any deviation from a convex object will add to that value. Using our algorithm, it is easy to identify whether a game object has holes or cuts that can affect its behaviour. The numerical value obtained can simply be integrated into further computations; for instance, the bounce of a ball in a certain direction, or its speed when rolling down a hill (see Figure 6).

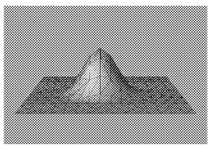


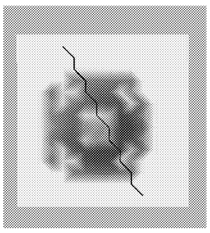
(a) Smooth sphere TAC = 12.566372 (b) Cut sphere TAC = 21.923119

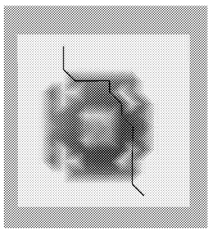
Figure 6: Spheres with different roundness

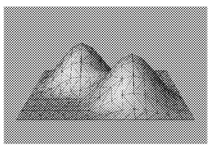
Other applications discussed here are related to virtual terrains. Terrains are very important in many computer games, as they provide the foundation of the digital worlds. Curvature characterisation of terrains give us a very suitable information for describing types of terrains. Indeed, a terrain can be viewed as an irregular surface, and it is possible to determine areas of the surface with similar values of curvature, by grouping together vertices of the same type (convex, mixed or saddle). Areas of positive curvature would represent mountains or valleys, while areas with mixed curvature are rugged terrain. The border between mountains and valleys would be composed of saddles, and thus have a negative curvature. This information can be used for terrain reasoning in 3D action games [Van der Sterren 2001], for example, for path finding on a terrain, which satisfies certain requirements. One requirement might be to minimise the energy that a game object needs to apply to navigate on the surface, or that the path chosen should go over the tops of mountains.

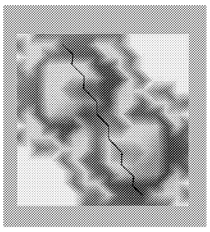
Some experiments have been carried out on the use of the curvature computed at each vertex of a surface. By trying to remain within regions of negative curvature, new paths can be found, which avoid obstacles and navigate around them, as shown in Figure 7. Here, regions of different curvature are coloured in red for positive, blue for negative, and green for mixed. For the two different terrains shown, two paths are considered, the straightest possible one, going from vertex to vertex, and another that considers curvature. This second approach can produce longer paths, but avoids going through the peaks of the mountains, thus having less climbs and descents, and a smaller change in height (Δh).


(a) One hill


(b) Length: 5.18, Δh: 1.99


(c) Length: 4.96, Δh: 0.91


(d) Two hills


(e) Length: 5.33, Δh: 3.01

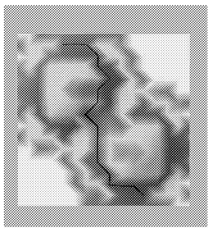
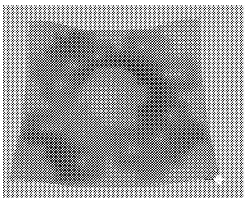

(f) Length: 5.99, Δh: 2.25

Figure 7: Path finding using curvature

Using curvature we will also define some other measurements for properties of a surface, for example the *Roughness* of a terrain surface can be defined, either in terms of the Gaussian curvature (i.e computing only the angle deficits of vertices) or in terms of all incorporated curvatures by applying the Polyhedral Gauss Map method. As seen before, both measurements identify regions of positive and negative curvature. The Roughness is obtained as the ratio of the total value for the positive curvature over the total value of the negative curvature, when the negative curvature is larger. The opposite ratio is taken when the positive curvature is larger. As a surface becomes more rugged, its Roughness will approach the value of one. This is illustrated with an example in Figure 8. The values presented clearly indicate that the Polyhedral Gauss Map method describes this feature more precisely and accurately. Other similar parameters can be derived by using curvature measures.

Conclusions and further work

The method presented to compute curvature of polyhedral objects, using Gauss Map, can provide extra information about objects and surfaces in a video game. The data obtained provides a good representation of the roundness of objects, and could be applied to improve the realism of the physical simulations.


(a) Deficit = 0.008
Gauss map = 0.090

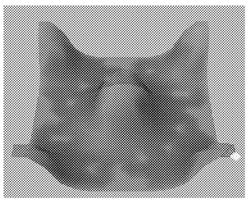

(b) Deficit = 0.033
Gauss map = 0.273

Figure 8: Roughness of two different surfaces

The algorithm is specially adequate for the kind of polygon meshes used in games, since it works for discrete polyhedral structures, and can cope with several different cases of vertices.
It still remains to implement these computations into an actual game physics engine, but the possibility offers good potential, as the algorithm proposed would not interfere with other aspects of a game, in a noticeable way.

REFERENCES

[Alboul and Echeverria 2005] Alboul, L. and Echeverria G. Polyhedral Gauss Maps and Curvature Characterisation of Triangle Meshes. In: Bez, H., Martin, R., and Sabin, M.(eds.), *The Mathematics of Surfaces XI*, pp. 14-34, LNCS 3604, Springer 2005.

[Kühnel 2002] Kühnel, W., Differential geometry. Curves-Surfaces-Manifolds, *Amer Math Society*, 2002.

[Meyer et al. 2003] Meyer, M., Desbrun, M., Schröder, P., and Barr, A. H., Discrete differential-geometry operators for triangulated 2-manifolds. In: Hege, H.-Ch., and Polthier, K. (eds.), *Visualization and Mathematics III*, pp. 35-59, Springer-Verlag, 2003.

[Peng et al. 2003] Peng, J., Li, Q., Ja Kuo, C.-C., and Zhou, M., Estimating Gaussian Curvatures from 3D Meshes. In: Rogowitz, B.E., Pappas, Th. N. (eds.), *Human Vision and Electronic Imaging VIII*, Proc. of SPIE 5008, pp. 270-280, 2003.

[Van der Sterren 2001] Van der Sterren, W., Terrain Reasoning for 3D Action Games,*2001 GDC Proceedings*, 2001.

[Quize 2003] Quize, K. S., Video Games in Education, IJIGS, vol.2, no. 1, pp. 49-62.

[Yamauchi et al. 2005] Yamauchi, H., Gumbold, S., Zayer, R., and Seidel, H.-P., Mesh Segmentation driven by Gaussian Curvature, *Visual Comput.*, 21, pp. 659-668, 2005.

Issues in the Physics of a Motocross Simulation

Benoit Chaperot, Colin Fyfe,
School of Computing,
The University of Paisley, Paisley, PA1 2BE, SCOTLAND.
email: benoit.chaperot, colin.fyfe@paisley.ac.uk

Abstract

In this paper, we investigate the use of a rigid body simulation engine to simulate a motocross bike. We discuss the movement of bikes from a theoretical perspective and apply this perspective to a simple motocross bike. This initial model suffers from control problems and we introduce several extensions which lead to a stable and enjoyable game. We believe that we have created the first realistic motocross simulator (as opposed to arcade game).

1 Introduction

Rigid body simulation, or physics simulation, is a method for simulating mechanical systems. It is generally present as a piece of software (library), used as part of another piece of software (in this case a video game).

Motocross Madness is a motocross game, released in year 2000; it is a popular and fun game to play. It does not make use of rigid body simulation. Other games, like MX Unleashed have been released since (see [10] and [9]), are nearly equally fun to play, and make use of rigid body simulation. However the simulation is used not to make the control of the bike more realistic, but to make the animation more attractive to the eye. One can see the rider moving on the bike, and the suspensions working as the bike goes over bumps, but the bike handles in a unrealistic way. These are called arcade games, as opposed to simulator games. In arcade games, fun and game play is preferred to simulation and realism.

The main difference between an arcade racing game and a simulation racing game is that in an arcade racing game the behavior of the vehicle is controlled by a set of rules, procedures and animations, whereas for a simulation the behavior of the vehicles is controlled by the physical and mechanical properties of the vehicle, and by the physics engine.

Simulators can offer a different, still enjoyable gaming experience, as proved by the success of games like Gran Tourismo. The use of rigid body simulation for vehicle simulation is not new. The library used for vehicle simulation can either be:

- A general purposes rigid body simulation library (for example [6], [11], [13], [12] or [4]); this solution has the advantage that it gives a lot of freedom and flexibility to developers to experiment and implement realistic vehicle simulation.
- A vehicle simulation library ([5] or [1]); this works well since the library is dedicated to vehicle simulation; it makes development easier; however, this solution may not offer as much flexibility as when using a general purposes rigid body simulation library.
- In house solution. This offers the most flexibility; however it may be very costly in terms of development.

There is no motorbike simulator on the market; this, together with a strong interest in bikes, simulators and video games, is our main motivation for creating what can be considered as the first motocross simulator.

In this paper, we investigate the use of rigid body simulation, to simulate a motocross bike, in the context of the Motocross The Force game. Motocross The Force is a motocross game featuring terrain rendering and rigid body simulation. An example of it in use can be seen at

`http://cis.paisley.ac.uk/chap-ci1`

The game has been developed and is still being developed in conjunction with Eric Breistroffer (2D and 3D artist). First, we have an overview of the theory behind bike riding, then we detail our approach to using rigid body simulation to simulate a motorbike and some of the improvements made over the original models. Finally we conclude by discussing the right trade off between simulation and arcade methods.

2 Riding a bike: theory

First, it is worth noting that a motorbike is by nature stable; this means that, in normal conditions, while riding a motorbike, the handle bars and front fork do not oscillate; the bike tends not to lean to one side, and it tends to go in a straight line.

There are two main things that make bikes naturally stable:

- Gyroscopic precession: this is a phenomenon occurring in rotating bodies in which an applied force is manifest 90 degrees later in the direction of rotation from where the force was applied.
- Rake angle and trail: as described on The Master Strategy Group site [2], wobble and weave are diminished because, when the wheel is pointing at an angle other than straight ahead, the contact patch is not in alignment with the direction of travel of the

bike, that is, a slip angle is created. A restoring force is applied to the contact patch by the ground which attempts to force that alignment. Thus, because of trail, the front wheel tries to go in a straight line (see Figure 1).

In practice, a rider can ride a motorbike with no hands, except to operate the throttle.

Let's now consider the main forces acting on the motorbike; for simplicity it is assumed that the rider is attached to the bike, and bike and rider can be considered as one body. The three main forces acting on the bike are:

- The weight, acting down: $F_g = mg$ with **m** the total mass, and **g** the gravity.
- The centrifugal force, acting horizontally, directed towards the outside of the turn:

  $$F_c = \frac{mv^2}{r} \tag{1}$$

  with **m** the total mass, **v** the linear velocity, and **r** the turn radius.
- The contact force, from the contact between the tyres and the ground.

The sum of these three forces can be assumed to be zero and the triangle of forces closed, if the bike is balanced.

Other forces, including traction, inertia, air friction, gyroscopic precession are also acting on the bike, but these can be ignored for now for clarity, mainly because these forces have no effect or no negative effect on the balance of the bike.

From Figure 1, and with the concept of these three main forces acting on the motorbike, one can assume that what makes a motorbike turn, is not the direct action on the handle bar; instead, it is the angle the bike is making with the ground (roll angle).

Besides, one can notice that the only force that can be controlled by the rider is the horizontal component of the ground contact force. This horizontal force, which acts at ground level, determines the roll angle the bike is making with the ground and is obtained by action on the handle bars.

As an example, let say the bike is going in a straight line; the rider wants to turn right:

1. The rider would first counter steer left, in order to pull the front wheel contact patch to the left, and put the weight of the bike to the right of the contact patches.
2. The weight makes the bike lean to the right.
3. The rider would then gently steer right, to close the triangle of forces, and balance the bike; the bike can be assumed to be balanced when the sum of weight and centrifugal forces lie in between the two contact patches.

The procedure is reversed if the rider wants to turn left or wants to go back to a straight line.

We are confident that most of the mechanical phenomena described here can be reproduced using physics simulation.

3 Physics Simulation model

Open Dynamics Engine (ODE) [6] was chosen for the simulation, because it is an open source project, with a large community of users maintaining it and enhancing it. Also it is probably as fast, accurate and stable as expensive commercial rigid body simulation packages. Open source gives the possibility to modify the engine and add the features that are missing for a particular requirement. LUA is a script engine which is used for scene creation. It allows modifying the simulation scene without having to recompile the executable. It allows the separation of code and scene data.

3.1 Collision Model

On a car simulation, wheels are often modelled as spheres. This is not appropriate in the case of our motorbike simulation, because as the bike makes an angle with the ground, the effective radius of the bike would reduce. Other primitives have been tested, such as thin cylinder primitives (disks), but these primitives proved not to be very stable. Another problem with thin disks is that because of the limited time step used for the simulation (1/100 of a second), a thin disk can fall through a plane, if lying against this plane. With this limited time step the simulation engine does not handle collision between primitives with limited thickness well (due to gravity, a very thin object can travel more than its thickness during one time step, and the collision with a coplanar plane can be missed).

For the rendering, a terrain engine is created. All vertices are positioned on a regular grid (horizontal x and y coordinates), but with different height (z coordinate). This makes the creation and editing of terrains easy and efficient because all the terrain can be created, edited, and stored in memory as a two dimensional height map. A triangle mesh collision primitive could have been used, but it would not take advantage of the two dimensional distribution of vertices, hence would not be as efficient as a collision primitive that will take advantage of this particular distribution.

For these reasons, two collision primitives have been created for ODE. One primitive is a cone, with high radius to length ratio, which is used for the wheels. The other primitive is a terrain, making full use of the particular distribution of vertices.

One problem with creating new primitives is that for each primitive, a call back function is needed for collision with any other primitive. Hence the number of collision functions grows exponentially with the number of primitives. For the terrain the problem has been solved by considering the collision of primitives with the terrain as collisions of primitives with the planes and edges making the terrain; hence reducing the collision problem as reusing all the other primitives collision functions with planes and rays. For the cone, only two collision functions have been implemented, collision between cone and plane, and collision between cone and ray, in order for cones to collide with terrains. For any other collision, the sphere collision functions are used; this proved not to be a problem and any discrepancies are not noticeable in the vast majority of cases.

3.2 Dynamic Model

Dynamic bodies are used for the simulation. Each dynamic body can be associated with one or more collision objects. More than one collision object can be used on the same body to create a simulation object with a complex collision shape. Rigid bodies have mass and inertia tensors; these masses and inertia tensors can be set independently from the shape of the associated collision objects. As an example, wheels are set to have cones as collision object, but have hollow cylinders for mass and inertia tensor. As a first attempt, a fork and front wheel have been modelled using ODE joints. The wheel was attached to the fork using a Hinge joint, and the fork is attached to the bike frame using a Hinge2 (suspension and double rotation joint). This proved not to be successful; because of the limited time step and corresponding lack of accuracy of the physics engine, each joint introduces an error, and the sum of the two errors means the front wheel looked as if it was very loose. Instead, a Hinge2 joint has been used to attach the front wheel directly to the frame; this proved to be more successful but removes the trail as discussed above. The fork is attached to the frame, using a Hinge joint and is given the same rotation as the front wheel Hinge2 top axis. The rear wheel is also attached to the frame using a Hinge2 joint. The rider's trunk is attached to the frame using a Fixed joint, to prevent him from falling off the bike. All riders' articulations, elbows, knees, wrists, are modelled using Universal, Hinge, or ball and socket joints, with limits set on the joints to prevent the rider doing forbidden moves, and allow him to be animated by the simulation in a realistic way. All simulation objects' size, position and orientations, and joints' position and orientation are obtained procedurally, from the rendering meshes. The user has the possibility, through LUA script, to choose the collision primitive for each object, set joint types, and modify objects' sizes, positions and orientations, and joint positions and orientations, masses and inertia tensors.

An AMotor (angular motor) is used on the rear wheel, to allow for bike acceleration and braking.

All masses, inertia tensors, and mass parameters are set experimentally.

The player controls are the direct action on the handle bar, and the torque applied on the rear wheel.

3.3 Initial Results

All tests and experiments have been carried out using a Evaluation Panel composed of three regular gamers; these regular gamers evaluated the game at every stage of development and fed back to the main game developer criticisms and comments. To ride a bike, the user has 2 pairs of controls used to turn left/right or accelerate/decelerate the bike. The bike is extremely difficult to control.

1. It keeps on falling onto its sides; it is nearly impossible to control the balance of the bike by direct action on the handle bar.
2. It keeps on flipping while accelerating or decelerating; it is difficult to find the right angular motor parameters (target angular velocity and torque to achieve acceleration and deceleration).
3. The biker seems very rigid, because the rider is glued to the bike seat.

4 Improving the simulation

A few improvements to the original models have been made in order to improve the simulation.

4.1 Indirect action on the handle bar

What the player really wants is to turn left or right; turning left and right is not achieved by direct action on the handlebar; instead, as described above, it is the angle the bike is making with the ground, that makes the bike turn, and this angle is obtained through action on the handlebar. Hence, as an experiment, let us interpret the left/right player control of the bike as a target roll angle the bike is to make with the ground; and let the game engine evaluate the appropriate action to apply on the handle bar in order to achieve this target angle. As a first consideration, we can state that the action on the handle bar is dependant on the bike's velocity; a rider does not turn the handle bar as much while riding at high speed than while riding at low speed. We also state that the action of the handle bar is dependant on the difference between the current bike roll angle and the target bike roll angle, and also dependant on the current bike roll angle. As an experiment, we test with the following rotation for the handle bar:

$$R_h = \frac{C_s * (A_c - A_t) + C_n * A_c}{v} \tag{2}$$

With $\mathbf{A}_c$, the current bike roll angle, $\mathbf{A}_t$ the target bike roll angle (mapped as left/right control of player), $\mathbf{v}$ the linear velocity, and $\mathbf{C}_s$ and $\mathbf{C}_n$ two parameters to be determined experimentally.

After a few tests to find appropriate values for $\mathbf{C}_s$ and $\mathbf{C}_n$, this proved very successful; the bike does not fall onto its sides anymore and balance is maintained. However, it is still difficult for the player to fully control the direction of the bike.

4.2 Adding a force

As an experiment, instead of using an angular motor for the rear wheel, a force is applied directly on the bike frame. Applying directly a force is appropriate because it conveys the feeling of a continuous thrust one can have while riding a motocross bike. This feeling of continuous thrust is in practice mainly due to the loose traction between the bike wheels and the ground.

This proved to be very successful; the bike was not flipping anymore at acceleration and deceleration. Beside, by changing the position where this force is applied, it is possible to get the front wheel to rise while accelerating hard, and the rear wheel to rise while decelerating hard.

4.3 Adding torques

To make the bike even more stable, as an experiment, the bike frame is attached to the static environment using an angular motor. For the three bike axes, target angles are set for the frame in relation to the static environment (ground), and the angular motor applies torques to the frame in order to achieve those target angles. Spring and damping can also be adjusted on this joint, in order to obtain the right joint behavior. The ODE AMotor joint had to be modified, in order to allow an object to be attached to the static environment, and to accept angles outside the $\{-\pi, \pi\}$ range. This proved extremely successful; the bike was a lot more stable, and it was easier to turn. This also allowed for one extra control, lean forward or backward (biker weight on the front or on the rear of the bike). More details about AMotors can be found in the ODE use guide [8]:

4.4 Detaching the rider from the motorbike

A new joint called linear motor, has been implemented. It is very similar to the angular motor, but works with forces and translations instead of torques and rotations. The rider trunk is attached to the bike frame using this joint, and the fixed joint is removed. This proved very successful. The rider with his body weight is now able to absorb part of the shocks, just as a real rider would. The simulation looks more realistic, and the bike is more stable. More information about creating new joints in ODE can be found in this paper [7].

4.5 Simulating trail

As seen above, because a Hinge2 joint has been used, there is no more trail to force the front wheel in alignment with the ground. A trail can be simulated by forcing the front wheel in the direction of the moving ground. The rotation for the handle bar now becomes:

$$R_h = \frac{C_s * (A_c - A_t) + C_n * A_c + C_y * A_y}{v} \tag{3}$$

With $\mathbf{A}_c$, the current bike roll angle, $\mathbf{A}_t$ the target bike roll angle (mapped as left/right control of player), $\mathbf{A}_y$ the current bike yawl angle, or difference angle between the forward direction of the bike frame, and the velocity vector of the bike. $\mathbf{v}$ is the linear velocity. $\mathbf{C}_s$, $\mathbf{C}_n$ and $\mathbf{C}_y$ are three parameters to be determined experimentally. This also proved to be successful as it made the bike even more stable.

5 Conclusion

Simulating a motorbike is more difficult than simulating a car, because a lot more bodies, joints and mechanical phenomena are involved. We believe that this is the first time such a motocross simulator has been successfully created.

Simple simulation models can make a bike realistic but makes the game totally unplayable. Modifying the models, to make the game playable may involve introducing controls and objects which are unrealistic. The work is not finished yet, but so far the simulated bike seems realistic, and the game seems fairly easy and fun to play.

The game is slightly harder to play than most arcade motocross games, but making the game easier would mean adding more unrealistic controls, cutting on the simulation, and the game would loose some of its appeal. It is also a choice to have the game appeal to a large public, and not only to young children. Current and future work involve fine tuning the simulation, and creating more bikes.

We [3] have also trained artificial neural networks using backpropagation and evolutionary algorithms to learn to ride such bikes. Future work will also investigate whether other computational intelligence techniques can be used for this purpose.

References

[1] http://www.carsim.com/. Technical report, Mechanical Simulation, 2005.

[2] C. Anthony and J. Davis. http://www.msgroup.org. Technical report, The Master Strategy Group, 2005.

[3] B. Chaperot and C. Fyfe. Motocross and artificial neural networks. In *Game Design And Technology Workshop 2005*. TBA, 2005.

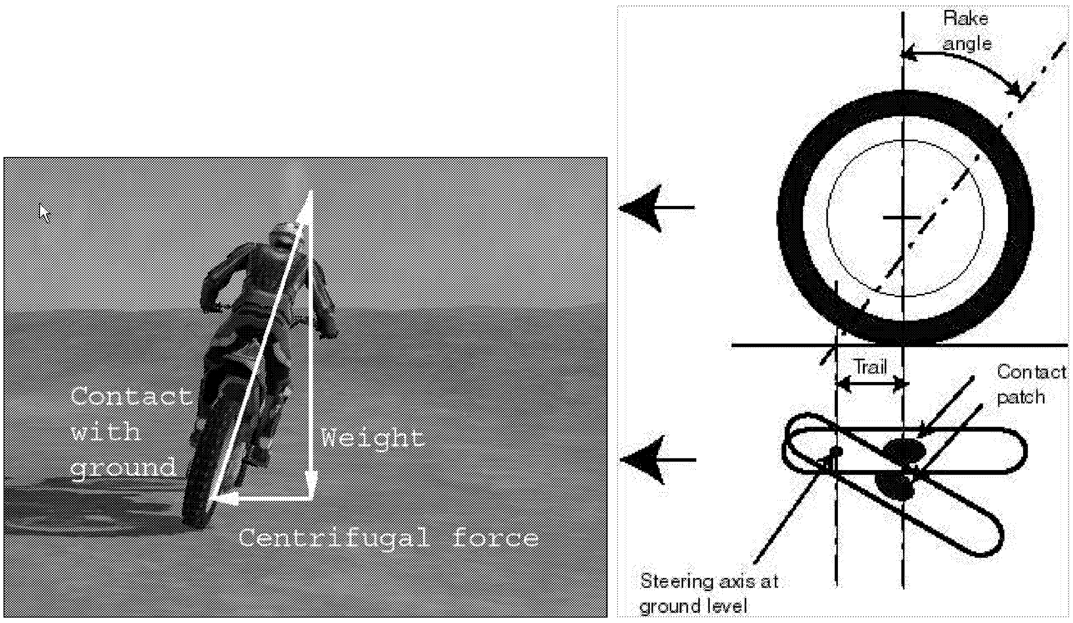


Figure 1: The three main forces acting on the bike. Rake angle and trail are used to make the bike stable, figure used with permission of The Master Strategy Group http://www.msgroup.org

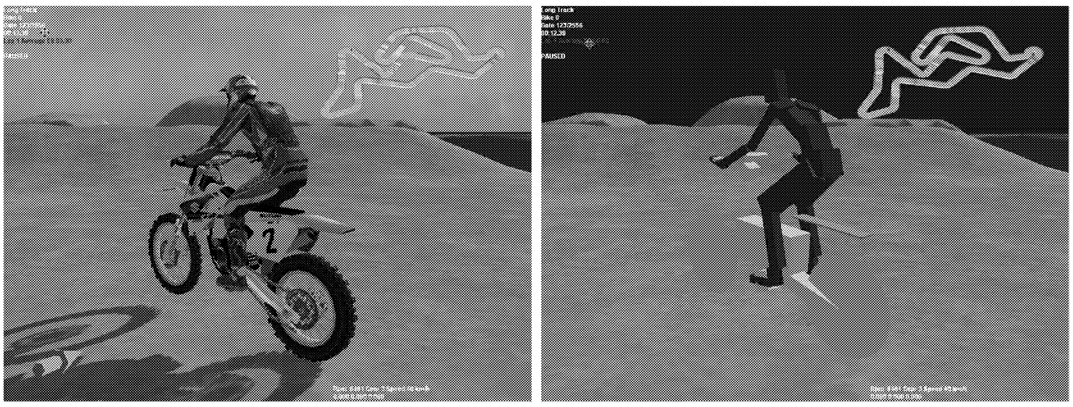


Figure 2: The bike has seen in the game and its associated collision objects used for the simulation

[4] David Lam. http://www.tokamakphysics.com/. Technical report, 2005.

[5] E. Laptev. http://www.oxforddynamics.co.uk/. Technical report, Oxforddynamics, 2005.

[6] R. Smith. http://www.ode.org/. Technical report, 2005.

[7] R. Smith. http://www.ode.org/joints.pdf. Technical report, 2005.

[8] R. Smith. http://www.ode.org/ode-latest-userguide.html. Technical report, 2005.

[9] Various. http://ps2.ign.com/articles/445/445638p1.html. Technical report, IGN, 2003.

[10] Various. http://thq.com/game.asp?1052—46045. Technical report, Rainbow Studios, 2003.

[11] Various. http://www.havok.com/. Technical report, Havok, 2005.

[12] Various. http://www.novodex.com/. Technical report, AGEIA Technologies, 2005.

[13] Various. http://www.renderware.com/physics.asp/. Technical report, Criterion, 2005.

PRODUCING ANIMATIONS FROM 3D FACE SCANS

Alan Robinson, Marcos A Rodrigues and Lyuba Alboul
Geometric Modelling and Pattern Recognition Group
Materials and Engineering Research Institute
Sheffield Hallam University, Sheffield S1 1WB, UK
{A.Robinson, M.Rodrigues, L.Alboul}@shu.ac.uk

KEYWORDS
3D face scanning, Structured Light Scanning, Facial Animation.

ABSTRACT

In this paper we take our existing research into 3D surface scanning using uncoded structured light, optimized for human faces, and develop a method to record a sequence of face movements and visualise these as an animation in real time; we call this method the *3D Animation Processor (3DAP)*. The applications for this work include 2D and 3D face recognition, broadcast and feature film animation, and computer games production. It should be stressed that the 3D recording of facial movements in real time is a difficult problem that is attracting considerable research attention; but accurate and appealing results have so far proved elusive. We record the faces of a number of speaking subjects, process the data representing their shape and colour as it changes over time, and visualise the animated face models. We identify a particular problem in *frame continuity*, whereby unacceptable jumps and jitters occur in both the shape of the face and its colour mapping, and begin to solve this problem using hole-filling and interframe interpolation. We also investigate methods of feature tagging, so that the model can be placed in a fixed coordinate system and thereby incorporated into computer generated animations.

INTRODUCTION

Current research on facial animation from live recordings is incipient. Computer generated animation often uses live recordings – such as from video footage which is then mapped onto the face, or when the keyframe artist refers to the action of a real person, when drawing the figure. However, these live recordings are nearly always in 2D, in the pixel space of a computer or the raster scan of a video signal. To use 3D recordings, in other words to have a 3D model of the face which changes at a suitably fast frame rate, requires firstly a robust 3D recording system, and secondly the methods and strategies required to incorporate the recordings in the finished work. This work may be an interactive computer game, a broadcast TV production, or a feature film, all of which media are looking to increase the liveliness and realism of their work.

These two requirements, a robust 3D recording system and a set of methods to incorporate the recordings into media productions, are the subject of a research programme in the Geometric Modelling and Pattern Recognition group, at Sheffield Hallam University, in the fields of animation and face recognition. In this paper we will discuss the application of 3D recordings for facial animation, and the main problems that have been discovered and are being addressed. The purpose of such research is to produce an appealing and lifelike rendition of the subject while maintaining temporal coherence. In computer games this needs to be achieved in an interactive environment, which means that the 3D model must maintain its shape and texture if its pose is changed by the player. A number of limitations exist such as the problem of achieving automated animation on a per frame sequential basis using information from current and previous frame. A common source of error is the general under-constrained "correspondence problem" which has been tackled through a number of techniques in computer vision with limited success (M.A. Rodrigues and Y. Liu. 2002), (Y. Liu and M.A. Rodrigues 2002), (Y. Liu, M.A. Rodrigues, Q. Wei, 2002). Extensive manual correction is required to compensate for errors and improve temporal coherence. Furthermore, traditional techniques such as streak-lines, or squash and stretch deformation require extensive temporal analysis for applications to video.

To solve the problems arising from the 3DAP method we create a sequence of 3D face models by straightforwardly adapting the methods for single scans, and then visualise this sequence as an animation of a talking face. Typical of these problems are hole filling, registration and texture mapping (Lu and Jain, 2005), which can all be included in a more general issue that we call *frame continuity.* Frame continuity can be addressed either in the input video frame, which is a 2D discrete pixel space, in the 3D surface mesh, which is a graph of 3D vertices, or in the rendered model, which is a multi-valued space including polygonal surfaces and colour values.

RIGID 3D SCANNING USING UNCODED STRUCTURED LIGHT

Our existing research into 3D scanning uses a novel "uncoded structured light" method, which projects a pattern of evenly-spaced white stripes onto the subject, and records the deformation of the stripes in a video camera placed in a fixed geometric relationship to the stripe projector. Figure 1a shows a detail from one such video frame, clearly showing the deformed stripes. The advantage of this over stereo vision methods is that the stripe pattern provides an explicitly connected mesh of vertices (Figure 1c), so that the polyhedral surface can be rendered without the need for surface reconstruction algorithms. Also, a smoothly undulating and featureless surface (such as in Figure 1) can be more easily measured by structured light than by stereo vision methods. We will also see that these advantages for single frame scanning are even more important for sequential scanning, i.e. animation.

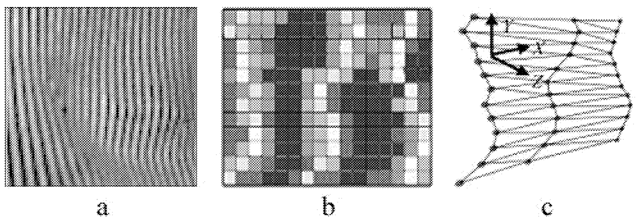


Figure 1. (a) Part of image showing stripe deformation. (b) Detail showing pixel array (c) resulting mesh of vertices.

Once the surface shape has been modelled as a polygonal mesh, we return to the video image, take the colour of the reflected white stripe at each pixel that maps to a vertex, and colour the vertex (or triangle) accordingly. The final model therefore contains the *(x, y, z)* coordinates and their corresponding RGB (*red, green, blue*) values for each vertex, and the face can be visualised as in Figure 2. This shows the original bitmapped image (the stripes are too fine to be discernible), and five arbitrary poses of the 3D colour-mapped 3D model.

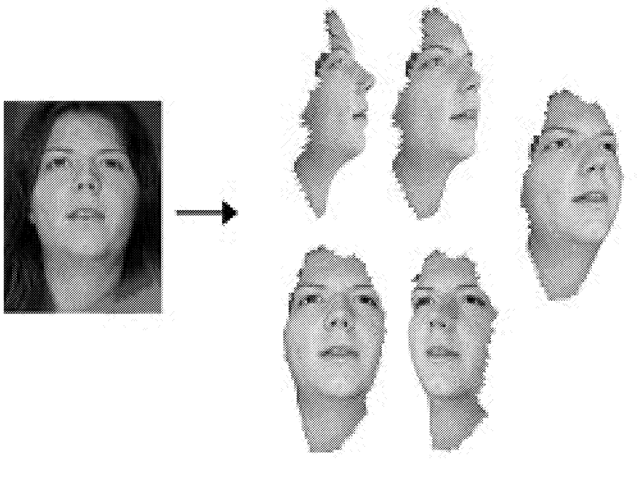


Figure 2. The input image (left) is transformed into the 3D model (right), showing five poses.

PRODUCING ANIMATED MODELS FROM VIDEO SEQUENCES – THE 3D ANIMATION PROCESSOR

To produce the video sequence, the system is set up as described above, and the video image received by the sensing camera is recorded as a sequence of standard PAL video frames at 25 frames per second. Typically the subject is speaking and an audio track is simultaneously recorded of their voice, so that the resulting 3D animation will be speaking synchronously.

Great care must be taken over the video recording device, and it was found that if the data is compressed as normally happens in an MPEG-2 recording, or if the video is recorded onto tape causing a loss of resolution and increased signal to noise ratio, then the bitmap image is degraded to the point where it may become unusable. In this work we record from a Canon XM1 video camera, through the Firewire IEEE-1394 output, into the capture software (iMovie) on an Apple computer. We then export the captured video as a sequence of .bmp files (24 bit, 768x576 pixels), which is the same data type as is used for the existing rigid scanner. Visual inspection and subsequent tests show that the image quality (in terms of stripe definition and bandwidth of the greyscale) matches very closely with the images from the rigid scanner. A detail of one frame is shown in Figure 3.

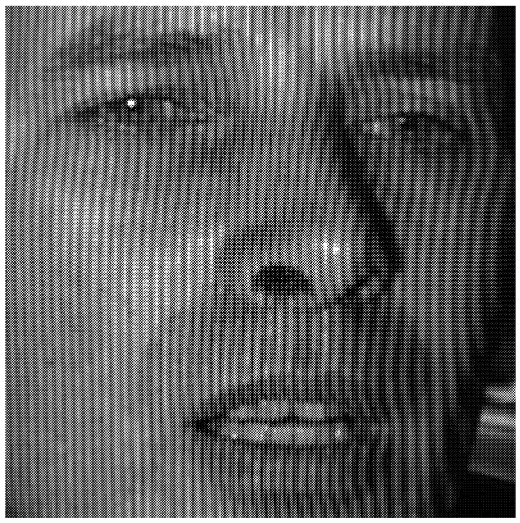


Figure 3. Detail of one frame in the recorded sequence, showing the deformed stripes.

Generating the 3D model from the sensed bitmap image requires our standard Indexing Method as described in (A. Robinson, 2005), (A. Robinson, L. Alboul and M.A. Rodrigues, 2004), followed by the output coordinates of the vertices (*x,y,z*) and their colour values (*r,g,b*) to our customised file format .rgb with six parameters for each vertex. This process is then semi-automated so that at a key press the next image file in the sequence is loaded, the indexing algorithm is performed, and the corresponding .rgb file is generated. This semi-automation allows the operator to compare the algorithms and resulting 3D model "live" as the process is happening, which reveals any problems in the continuity between frames. During this live operation, adjustment is provided for the following parameters: all intrinsic and extrinsic constants, stripe width and amplitude, illumination threshold, seed position, hole filling, and others.

TESTING AND INITIAL ANALYSIS OF FRAME CONTINUITY

The attributes to be tested in the animation sequences will of course depend upon the chosen application, but the measures that will give general indications of success are variance in geometric measurements between frames, and subjective appeal to the viewer. Although these appear to be two totally independent and quite distinct measures, they are in fact closely related; a sudden jump in say the position of the eye between two frames will be picked up in the variance results and will also be subjectively noticed by the viewer. In this regard the viewer may well be more reliable, as he or she will easily track the position of the eye and report any inconsistent movements, whereas the measuring system will firstly have to recognize the eye. Although recognizing the eye is a well-understood task, other features, even obvious ones such as the tip of the nose, are more difficult to find consistently, and so will make the variance report less reliable.

The first tests were conducted on a speaking person, who was scanned and texture-mapped by our 3DAP method. 113 frames were recorded, at 25 frames per second, of which six are shown in Figure 4. Figure 3 shows a detail of one frame. If we consider only a single frame, in other words a rigid body, then results show that all of these scans will work well, and using standard reconstruction and hole-filling algorithms will give a good model of the face. However, both data analysis and subjective inspection clearly show that for the moving subject, while there is good continuity from frame to frame within the surfaces, on the boundaries there are unsatifactory jumps between frames. For instance, the boundary of the nostril changes irregularly in shape, because the occluded area within the nostril is very sensitive to slight changes in pose; and whereas the pose itself is changing smoothly, the discrete nature of the pixel samples and of the segmentation algorithms combine to give unpredictable results. Similar problems occur around the lips, which in a speaking subject will inevitably be moving and deforming quickly and significantly. Inspection of the test data shows that the boundary of the lips, what one might call the inside of the mouth, is dependent upon the occlusions created by the mouth shape, and does not provide a consistent description of the lip shape. This in turn presents serious consequences if, say, a recording of the lip movements is to be incorporated into a computer generated model.

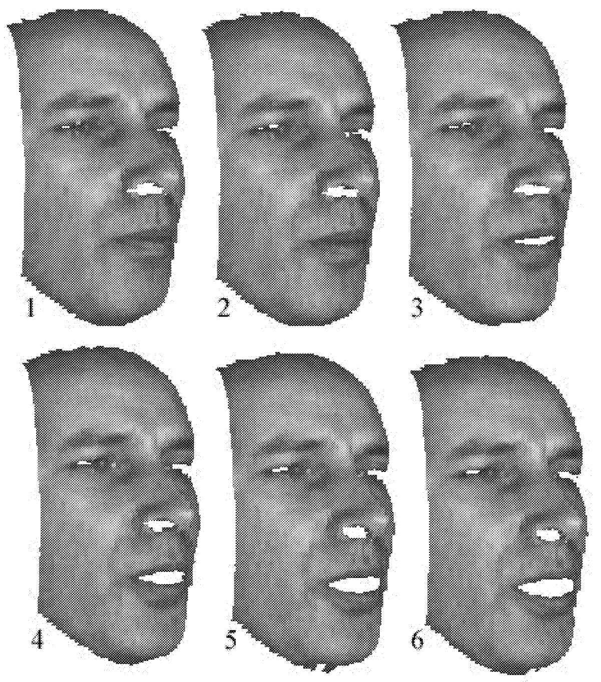


Figure 4. Texture-mapped model of six consecutive frames.

This in turn leads to the question of how we can define the edge of the nostril, and of the lips, chin, eyes, and any other features; because if we can do so then we can constrain the algorithms to give a better definition of the feature boundaries. This feature-finding is important in a number of scenarios: to control the edges of the holes and therefore improve hole-filling, to provide cleaner and more consistent edges to the model, and as a basis for registration issue, which will be dealt with in the Frame Registration by Feature Tagging section of this paper. Firstly we will look at the general issue of hole-filling.

FILLING HOLES

Structured Light scanning has the advantage over Stereo Vision methods in providing an explicit graph of the connections between vertices, which in our case is in a rectangular grid pattern formed by the vertical stripes. If every projected light element is reflected on the target surface, sensed in the recorded image, and measured as a surface point, then there will be a one-to-one correspondence between projected elements and modelled points. This also means that the graph of connections of the projected elements is homeomorphic, or bicontinuous, with the graph of connections in the modelled mesh. Therefore the mesh can be connected using the same pattern as has been projected. However, it is likely that some elements will be missing from this grid, maybe because a projected element misses the target surface, or is occluded from the camera view, or is rejected by the indexing algorithm, and we will call these *holes* in the graph of vertices. Additionally there are projector occlusions, where there are no missing vertices, such as may happen on the side of the nose. Here the graph connections do not apply, and so if we are to fill in the occlusions, we must add new mesh connections.

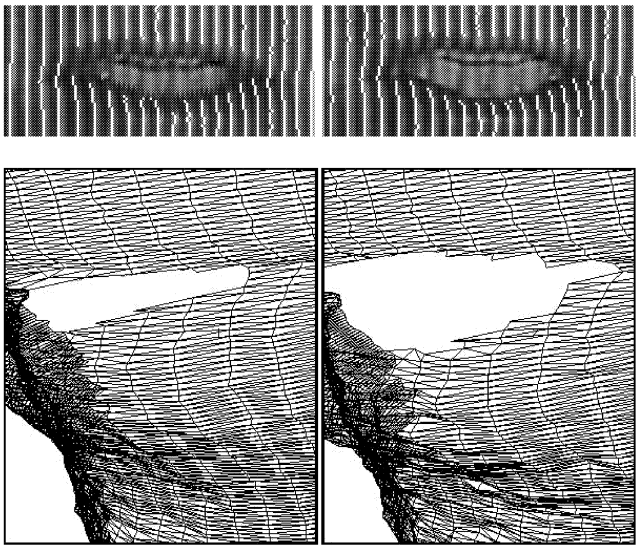


Figure 5. Model of two successive frames, showing detail of lips. Above is the original source image, with recognized stripes emphasised. Below is the resulting mesh visualisation.

Figure 5 shows the principles involved, for two successive frames shown left to right. At the top are details of the bitmap image showing the mouth slightly opening from the left frame to the right frame. The bitmap is similar to that seen in Figure 3, with the white stripes as recognized by the algorithms emphasised in the image. In this case the algorithm stops when it reaches the lips, but in an uneven manner due to the occlusions presented by the lips. The generated meshes shown underneath are derived *via* the mapping between each pixel on the white stripe above, and its corresponding vertex in the mesh below. It is clear that the uneven lip boundary is also present in the mesh, and over the course of the animated sequence this produces unattractive jitters. The problem here is that the viewer is very critical of temporal changes which do not flow smoothly, and will be disturbed by surface edges which jitter from frame to frame. One solution is to fill in the hole in a way which smoothly follows from frame to frame. The holes in the mesh correspond exactly to the missing parts of the stripes, so that if those stripes can be connected in some way, then the homeomorphism will allow the vertices to be similarly connected in the mesh (Remember that this does not include projector occlusions). This is a simpler and potentially more accurate alternative to surface reconstruction from the polygonal mesh, which is the more usual approach, and requires a 3D interpolation rather than the 2D interpolation which can be used here..

Hole filling methods (Tekumalla and Cohen, 2004), (Wang and Oliveira, 2003), or stripe connecting methods in our context, range from simply connecting the edges of the holes with straight lines and planes, to using curves such as cubic splines and Beziers, to our "hole-patching method" (see below in Figure 9) which replaces the hole with the patch on the symmetrically opposite side of the face. These techniques can work well for a rigid surface, but when the surface is moving the problem becomes much harder, because the interpolation of missing surface may be inconsistent from surface to surface.

For single frames, i.e. rigid surfaces, we currently use a simple straight line interpolation, which if the controls are set correctly will produce satisfactory results. So the first step here is to try the same method over a number of frames.

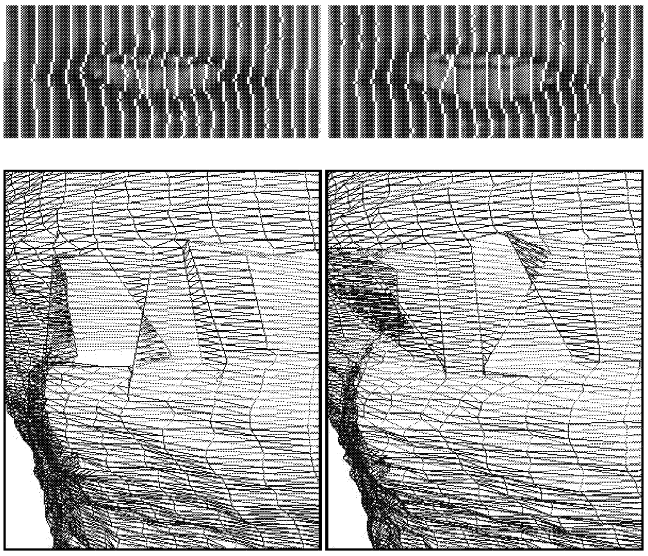


Figure 6. As in Figure 5, with automatic straight line interpolation across the mouth.

Figure 6 shows the same data, but with a simple automatic straight line filler added. This is a worst case example showing lines which almost meet, and in general an inconsistent spacing between the stripes. The problem occurs partly because the edge of the lips, which is the point at which the straight line should join with its continuation on the other side of the mouth, is irregular, and partly because there the teeth provide a specular surface which reflects the stripes in unpredictable ways. It has been found that even if the stripes are prevented from irregular changes of direction as shown here, if the spacing between stripes is not smoothly continuous, the results will be unacceptable due to interframe jitters.

Therefore our solution is to introduce some interpolation between frames, and to keep the spacing between the stripes as even as possible. The key issue here is that this solution is ***only relevant for the mouth,*** where a straight line between the lips gives satisfactory results. It is not appropriate for, say the side of the nose (see the Frame Registration and the Texture Mapping sections). This means that we must segment the face, approximately as shown by the rectangles in the Figures 5, 6 and 7, to constrain the hole filling region. The results of this interpolated straight line filling can be seen in Figure 7, which shows a much smoother and more consistent filling, which provides smooth results between frames. This will also allow a better colour mapping to be achieved, with the proviso as mentioned below that problems may occur if the interpolation causes a colour to map to the wrong colour.

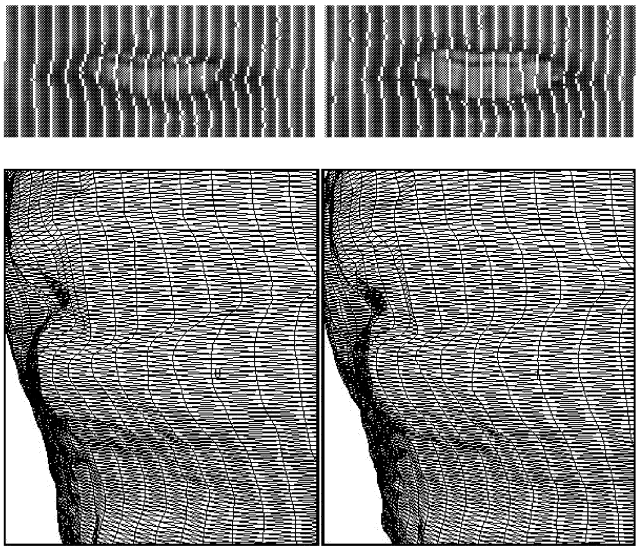


Figure 7. As in Figure 4, with interpolated hole-filling across the mouth.

despite this proviso, these results become even more appealing when colour mapping is applied to the surface, as in the example shown in Figure 10.

As has been said, this interpolation method is not suitable for the common occurrence of an occlusion at one side of the nose. Instead we assume that the other side of the nose will not be occluded, and find a symmetrical patch of shape and texture which can be reflected and used. Details of this are included in the next section, and are visualised in Figure 9.

FRAME REGISTRATION BY FEATURE TAGGING

The registration problem (M.A. Rodrigues, R. Fisher and Y. Liu 2002) is normally encountered when two overlapping patches modelling the same surface each have their own independent coordinate system. The task is then to join, or register, the two patches, by finding a transformation (translations and rotations) between the two local coordinate systems. In our 3DAP the problem is slightly different in that we are trying to register two or more patches which are separated in time, but have the same coordinate system. Therefore if we consider, say, a fixed point on the forehead of the subject, that will not have the same location from frame to frame if he or she is moving. We need a common coordinate system from frame to frame referenced from a fixed point on the face or head; the ideal candidate would be the skull, which of course is not possible!

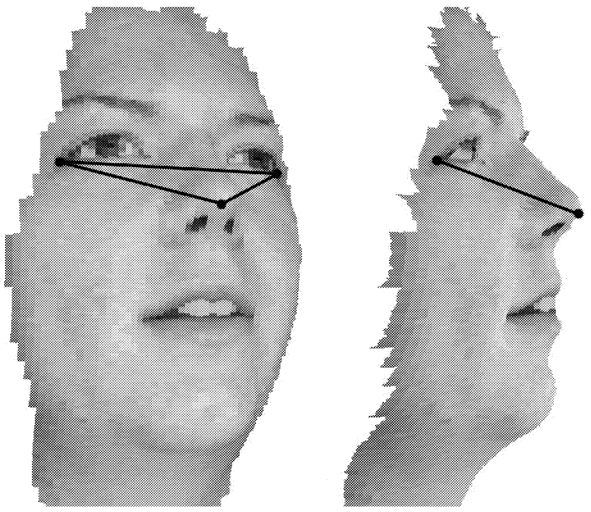


Figure 8. Two views of the tagging procedure.

We currently use a "tagging" method, as shown in Figure 8, by manually marking the outside corner of each eye in each frame. This can be done either in the bitmap image or in the polyhedral model, but we currently use the polyhedral model, because the 3D space is then used to find further features. These extra features are found automatically having tagged the corner of the eyes; the most important current one is the tip of the nose. Finding this has long been acknowledged as a difficult problem in face recognition - what exactly is the tip of the nose? One method is to use a convex hull, such as offered in Matlab as the function `convhulln`. This will give a small area around the tip, and the centre of mass can then be used to give a specific point. Another method is to describe an ellipsoid

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1 \qquad (1)$$

using the tagged eyes as foci, such that the distance *a* is from the left eye to the tip of the nose, *b* from the right eye to the tip of the nose, and *c* is minimised so that the ellipsoid just touches the nose at its tip. This is the method shown in Figure 8. Tests have shown that these methods will still give some jittering from frame to frame, partly because of the pixel sampling which models the surface discretely. Again the solution, which is under investigation, is likely to be interframe interpolation, to smooth out the location jumps.

The points are then used as the basis of a registration method, whereby the tip of the nose is taken as the origin of the face coordinate system, and the line from the origin bisecting the two eyes then forms the negative *Z*-axis. If frame continuity is achieved for these points, then the face, or more precisely the eyes and the nose, when animated, will remain fixed in the display.

There are a number of applications for frame registration, as suggested in the previous section. One is to enable patching of holes in the surface; a method which we are developing uses the *Y-Z* plane as a plane of symmetry, which enables us to fill holes in one side of the face with the corresponding patch in the reflected side of the face. A likely candidate is the hole which can occur on one side of the nose due to occlusions, which can often be successfully filled because the shape and texture of the nose is fairly symmetrical.

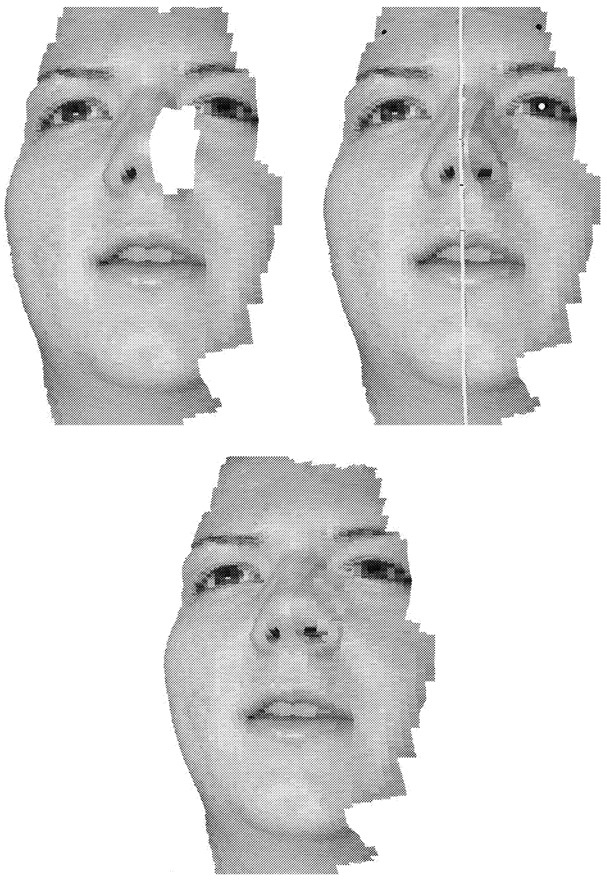


Figure 9. The patcher used to recover nose shape. Top left shows the occluded hole; top right shows the line of symmetry and filled patch; below is the finished model.

Figure 9 shows one such example; the top left image shows the occlusion on the (subject`s) left side of the nose. At the top right a tagging method similar to the one discussed above has been used to find a plane of symmetry at the centre of the face. A correspondence is then determined between vertices on the right and left half of the face, and a mapping is found between each missing vertex on the (subject`s) left half of the face, and its pair on the right half. In this application the structured light scanning method proves very useful, as the explicit mesh exists for all vertices, even if those vertices are missing in the polyhedral representation. However, if the missing vertices are the results of projector occlusions, new connections will have to be added, such as by Delaunay triangulation (implemented by the `delaunay` function in Matlab). The patch as filled is shown by the dark (green) area in the top right image, and the final model is shown in the lower image. Note that colour mapping has been included as well as shape. This method gives good results for single frames, but at the moment the issue of frame continuity has not been solved for this application.

An important application for frame registration is when the animated model is to be combined with a synthetic object produced by computer generated methods. For instance, if the black triangle seen in Figure 8 were replaced by a pair of spectacles, it should be possible, if there is sufficient frame continuity, for the animation of the person to be wearing the glasses in a consistent manner, no matter how much the head was moving.

An extension of this application, and probably the most important goal of 3D scanning, is to add even more feature points so that the whole face, or possibly just parts of the face such as the lips, can replace the synthetically produced feature in a computer generated model. This would give the CG animation the added realism of live action of, say, a persons lips moving in perfect synchronicity with an audio speech track. (See Future Work for more ideas on this subject).

COLOUR MAPPING

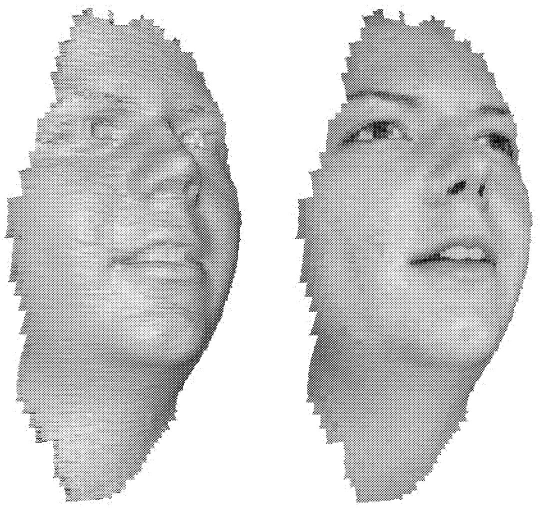


Figure 10. (left) face visualised as illuminated grey surface, (right) with colour mapping.

In the colour mapping process we decorate the surface, without making any changes to the shape. In the 3DAP process we take the colour from each pixel in the centre of each white stripe, and map that to the corresponding vertex in the polyhedral mesh. This colour mapping has advantages and disadvantages in terms of the 3D modelling of the face. Figure 10 (left) shows the triangles produced from the polyhedral mesh rendered synthetically (in this case using the OpenGL Lighting Model for a diffuse surface) as an illuminated grey surface, and Figure 10 (right) shows the same polyhedral mesh with colour mapped onto the triangles. Clearly the colour mapped surface has a greater appeal, although it is insignificant in terms of surface measurement. The advantage of colour mapping is that with very little effort it makes the face look more realistic, but it can then be very difficult to persuade the viewer that he or she is looking at a 3D model and not a 2D photgraph. We ask the reader to make their own judgement on this by inspecting Figure 10. It is then very difficult to dispel this thought that they are looking at a 2D photograph, unless there is some form of user interaction to prove the 3D-ness of the visualisation. Another problem with colour mapping is that the map will probably need to be distorted in order to fit the shape. In our scanning system the colour map is a single 2D image usually taken from the front, which means that surfaces that are oblique to the viewing direction will have apparently smeared or stretched texture, due to the horizontal scaling required; this can be seen in Figure 11 at the side of the nose. Taking multiple colour images from different directions would alleviate this effect. If the subject is carefully lit the colour map can be used inversely to modify the actual surface shape. Standard texture mapping methods such as "bump" and "displacement" mapping (James F. Blinn, 1978) can be used here. For successive frame animation the colour mapping process can again cause frame continuity problems, mainly when there is a hole in the surface.

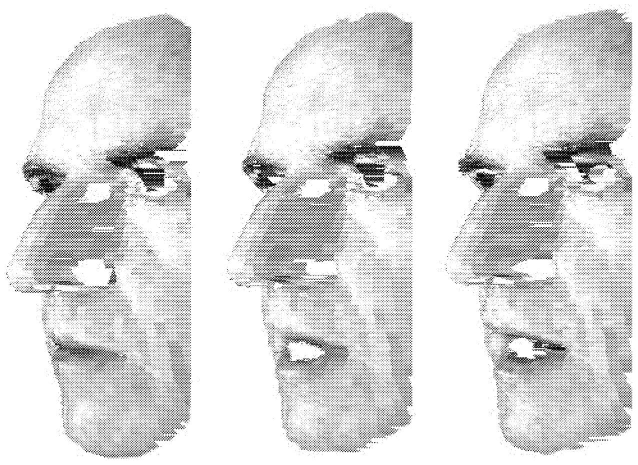


Figure 11. Texture mapping of three successive frames, without post processing. Occlusions around the nose and mouth, misregistering of the eyes, and streaking on the side of the nose, are clearly visible.

This effect is noticeable in Figure 11 around the eyes. If the hole is caused by an occlusion then no colour will be available at that part of the surface, and so interpolation will be difficult. The current method performs a bilinear interpolation of the colour starting at boundary edges, but this gives an undesirable plain colouring, and possible jittering. A better method is to use our "hole-patching method" to provide the colour map, although this is still subject to frame discontinuity.

It should also be noted that our 3DAP method can cause problems with the interpolation of the colour map, because the vertex has to correspond to the colour at the exact centre of the stripe. Therefore if the vertex is interpolated and moved to a new position, it may map, say, to a dark part of the stripe, and cause unwanted artefacts.

CONCLUSIONS

This paper presents an initial analysis of sequential 3D face scans, which is becoming an interesting and important area in face recognition and multimedia disciplines. We identify three typical issues – hole filling, registration and colour mapping – and report the investigations that we have undertaken to improve the key problem area, that of *frame continuity*. The analysis uses subjective viewing (relevant especially in multimedia applications), and variance in the surface measurement (relevant for face recognition and industrial applications). The initial approach that we have taken is mainly heuristic, one of identifying problems and trying various solutions. However, as our investigations have progressed we have identified two key issues that show a more logical progression between problem and solution. Firstly, it has become obvious that smooth frame continuity is extremely critical, both for shape and texture, and therefore an interframe interpolation must be used. Initial work shows that this will have a dramatic effect on subjective appeal. Secondly, both interpolation techniques and registration methods require at least a small amount of feature tagging; a minimum would be the eyes and tip of the nose. In an animation context this comparatively simple task can be achieved manually by the animator, but our work in face recognition shows that there are also possible methods of achieving this automatically.

We would like to stress that our novel method allows the capture of 3D motion in a sequence of video and then visualise the effects in real time. This is a significant advance over current methods with specific applications to the game industry. A computer game typically has a game engine for rendering objects and for managing sound and motion. Normally, a game engine is re-used across game adventures. Game designers use the tools in the game engine to create interactions between the player character and the various game objects. We argue that one important development of our techniques is the creation of specialised libraries, for instance, of realistic lip synchronised sequences that can be used across many games. The creation of such libraries is a straightforward process, and only requires the use of existing editing tools for subsequent integration into specific game sequences.

FURTHER WORK

We have identified a number of areas for further work including automatic feature tagging, colour mapping from multiple cameras, time-dependent interpolation for filling surface holes, and smoothing colour and shape jitters. An immediate task is to begin incorporating these live recordings into computer generated animations, and the work so far clearly demonstrates that a lifeliness is easily achieved by these methods which will enhance and speed up the work of the animator. We have performed distortion of the nose and overall face shape based on the tagged features, and will also add computer generated elements such as a motion captured body to the model. These investigations will begin to show how well the recorded and generated elements can be seamlessly mixed in the final output. The research reported here, despite the many problems yet to be solved, can already provide significant benefits to the games, film and TV community by including measurement and visualisation in time and in 3D.

ACKNOWLEDGMENTS

This research is supported by Sheffield Hallam University's Strategic Investment Fund (SIF) and Yorkshire Concept funding.

Many thanks to Willie Brink for his contributions to the hole-filling methods.

REFERENCES

M.A. Rodrigues and Y. Liu. 2002 "On the Representation of Rigid Body Transformations for Accurate Registration of Free Form Shapes", *Robotics and Autonomous Systems*, Volume 39, Issue 1, Pages 37-52, 30 April 2002.

Y. Liu and M.A. Rodrigues. 2002 "Geometrical Analysis of Two Sets of 3D Correspondence Data Patterns for the Registration of Free-Form Shapes", *Journal of Intelligent and Robotic Systems*, vol. 33, no. 4 ,(2002) 409-436.

Y. Liu, M.A. Rodrigues, Q. Wei, 2002 "Using Neighouring Relationships to Eliminate False Matches for Accurate Registration of Free-Form Surfaces", *Journal of Digital Imaging* Vol 15 (2002) pp 267-269.

A. Robinson, 2005 "Surface Scanning with Uncoded Structured Light Sources", *PhD Thesis*, Sheffield Hallam University, UK, 2005.

Xiaoguang Lu and Anil K. Jain, 2005. "Integrating Range and Texture Information for 3D Face Recognition", To appear in *Proc. IEEE WACV*, Breckenridge, Colorado, 2005.

A. Robinson, L. Alboul and M.A. Rodrigues, 2004 "Methods for Indexing Stripes in Uncoded Structured Light Scanning Systems", *Journal of WSCG*, 12(3), 2004, pp 371-378.

Lavanya Sita Tekumalla, Elaine Cohen, 2004. A hole-filling algorithm for triangular meshes. *Technical Report UUCS-04-019,* School of Computing, University of Utah (2004).

Wang, J., Oliveira, M. 2003. A hole-filling strategy for reconstruction in smooth surfaces in range images. In: 16th Brazilian Symp. on Computer Graphics and Image Processing, IEEE Computer Society (2003) 11–18.

M.A. Rodrigues, R. Fisher and Y. Liu 2002. "Special Issue on Registration and Fusion of Range Images", *Computer Vision and Image Understanding*, Volume 87, Issues 1-3, Pages 1-131 (July 2002).

James F. Blinn, 1978. "Simulation of wrinkled surfaces", *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, p.286-292, August 23-25, 1978.

GAME DESIGN

APPLICATION OF VOLERE SHELLS AS A PRINCIPLED APPROACH TO REQUIREMENTS CAPTURE AND TEST PLANNING FOR COMPUTER GAMES

Robert Clutton and Andrew Tuson
Department of Computing
City University, London
Northampton Square
London EC1V 0HB, UK
E-mail: robert.clutton@ntlworld.com, andrewt@soi.city.ac.uk

KEYWORDS
Software Engineering, Requirements Capture, Testing

ABSTRACT

Software Engineering is an issue of increasing importance to the games industry. This paper looks at the linked issues of requirements capture and testing, and argues that an adaptation of a modern requirements method – Volere Shells – is potentially useful to games developers as a lightweight but principled approach to addressing these issues. A simple case study will be used to illustrate the points made in this paper

INTRODUCTION

Computer game projects are increasing in size and complexity, driven by both ever more advanced technologies and increasing customer expectations. However, this means that such projects are increasingly vulnerable to expensive failure, and thus the use of more effective software development methods is needed.

To this end, this paper focuses on two of the key areas in any software development project; these are requirements capture and testing (Sommerville 2001). This paper shows how a modern requirements method – Volere Shells – is applicable to games development.

Software Engineering in Games

The waterfall model and prototyping methods seem to be applied across the industry but there are also examples of companies using agile methods such as extreme programming (XP), e.g. (Beck 2004), and other methods such as the unified process (UP) to develop their games and indeed a combination of all of the above.

On the other hand, there appears to be little literature, academic or otherwise, on software engineering in games and one main source we found were developer websites. To compound this, current practice is not well-studied or documented, an important first step to identifying issues in current practice that can be developed.

Otherwise the best literature on software engineering in games are books such as Object Oriented Games Development by (Gold 2004) and the Game Programming Gems series, e.g. (DeLoura 2000). However much of the content of these texts (e.g. object-orientated approaches and design patterns) can now be found on most modern computing undergraduate degrees.

The emphasis in software engineering practice in games appears not to be even. For example, testing and quality assurance in games is relatively well-developed. In fact the International Games Developers Association have produced "good practice" guidelines in this area (Rees and Fryer 2003).

In contrast, the requirements process is invariably an informal process derived from discussions with the creative team. As we will now discuss, this is a problem not only because clear requirements prevent possibly expensive misunderstandings as to what is to be built, but that also good requirements capture informs testing.

Linking Requirements and Testing

Requirements and Testing issues are extricably linked., even in "agile" methods such as XP (Beck 2004). XP is an increasingly popular approach in games development; it suggests placing a high emphasis on testing and evolutionary design at each iteration of the application. However, despite claims by extreme proponents of XP, for the test cases to be derived it must be clear what the system must do; this requires clear requirements, and thus some form of principled requirements capture.

One aspect of value from agile methods is the neccessity that the requirements process needs to be lightweight and easily adaptable to changes; in constrast with some traditional methodolgies for software development, e.g. SSADM (Ashworth and Goodland 1989), that largely assume that requirements will not change during development.

Rapid development and change in requirements is definitely the case in games development projects. For example, (Eberly 2004) stresses the importance of change control in the context of scene graph management:

"This aspect of frequent change is what makes software engineering for a game somewhat different than that for other areas of application. Knowing that change will occur as often as it does, you need to carefully architect the scene graph management system so that the impact of a change is minimal and confined to a small portion of the engine."

Therefore tools and methods are required that can achieve all of the following: act as an effective requirements capture process, be lightweight and thus adaptable to changing requirements, and help in providing a test plan that is easily kept consitent with these requirements.

Structure of this Paper

Given the above, this paper examines the application of a modern requirements engineering method – Volere Shells – with the aim of providing games developers with a principled and useful method.

This paper will first provide an informal introduction to Volere Shells and how they are used, and how they can be adapted to games development projects.

The case study used here to illustrate the application of Volere Shells is then described, a reimplementation of an arcade classic: Battle City. The application of Volere Shells to produced both requirements and test plans is then described.

Finally the results are put into context, conclusions made and future work discussed.

VOLERE SHELLS

The Volere Requirement Shell was developed by James and Suzanne Robertson of Atlantic Systems Guild and represents a lightweight template for requirements specification (Robertson and Robertson 2002). It is a way of storing a single requirement in a tabled format (see Figure 1) and can be used to reference other requirements within a project as well as content relating to that requirement.

The Volere process puts a strong emphasis on quantifiable requirements and also complements traceability and test planning processes. These templates are generic and it is encouraged to modify them as best suits the project. This means that this template is not only suitable for one industry, but it is suitable for many industries, including games.

Volere should not be considered to be part of a lifecycle process, but should be seen to complement any software process as a way of storing requirements and creating test plans.

As mentioned, Volere puts a strong emphasis on achieving quantifiable requirements by specifying a fit criterion. This is an objective measure of the requirement and this is what is evaluated in order to test whether the requirements have been met. All requirements must have a fit criterion and it should not be ambiguous, or include several requirements at one time. Each requirement must be specified separately and with separate fit criterion.

A requirement may contain dependencies or conflicts which this template will allow for. It will also allow for links to previous requirements and further literature relating to the requirement.

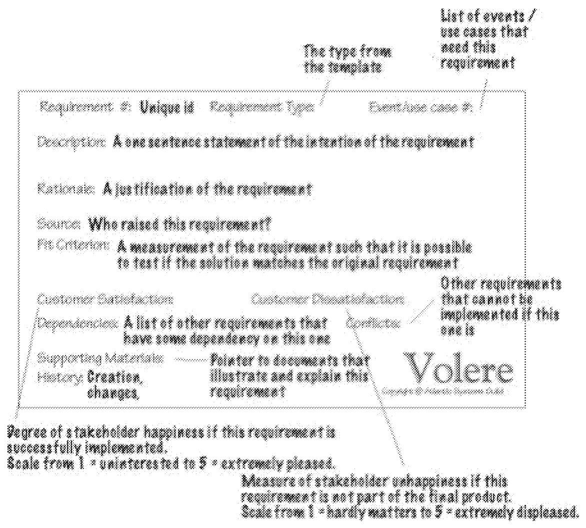

List of events / use cases that need this requirement
The type from the template
Requirement #: Unique id
Requirement Type:
Event/use case #:
Description: A one sentence statement of the intention of the requirement
Rationale: A justification of the requirement
Source: Who raised this requirement?
Fit Criterion: A measurement of the requirement such that it is possible to test if the solution matches the original requirement
Customer Satisfaction:
Customer Dissatisfaction:
Other requirements that cannot be implemented if this one is
Dependencies: A list of other requirements that have some dependency on this one
Conflicts:
Supporting Materials:
Pointer to documents that illustrate and explain this requirement
History: Creation, changes,
Volere
Degree of stakeholder happiness if this requirement is successfully implemented.
Scale from 1 = uninterested to 5 = extremely pleased.
Measure of stakeholder unhappiness if this requirement is not part of the final product.
Scale from 1 = hardly matters to 5 = extremely displeased.

Figure 1: An Example Volere Shell (Robertson and Robertson 2002)

The example project chose to implement the following structure within the template:

- Requirements Number: Requirement identifier
- Requirement Type: The type of the requirement (e.g. functional, performance etc)
- Event: An event that would trigger this requirement
- Rationale: The reason for this requirement to be included
- Source: Were this requirement was initially gathered
- Fit Criterion: A quantifiable description of the requirement
- Satisfaction: A measure of 1-5 of how satisfied the client would be if this requirement was met.
- Dissatisfaction: A measure of 1-5 of how dissatisfied the client would be if this requirement was not met.
- Dependencies: Any requirements that would have to be met for this to be achieved.
- Conflicts: Any requirement that conflicts with this.
- Supporting Materials: Any references relating to this requirement.
- History: any old requirements that this requirement may have evolved from.

Any project, regardless of its size and chosen methodology will contain requirements, games are no different in this respect. In many agile methods, keenly used in games, an extensive requirements and analysis phase are often overlooked. However, the Volere Shell offers a lightweight and simple solution to documenting requirements and producing relevant test scripts that are vital to any project.

THE CASE STUDY: BATTLE CITY

The case study used is taken from (Clutton 2005), and full details can be found there. The original Battle City had a playing field with different environmental elements on the field. The player(s) operated a tank against Non-Player Character (NPC) tanks. The objective of the NPCs was to search and destroy the players' tank and the players' home base, whereas the players' objective was to protect their base by eliminating the NPCs. Awem Studios have created their own shareware version of Battle City, Battleman, which performs in the same way as Battle City. You can see from the screen shot (Figure 2) there are several types of environments, brick walls, water and grass as well as various collectables that can be picked up.

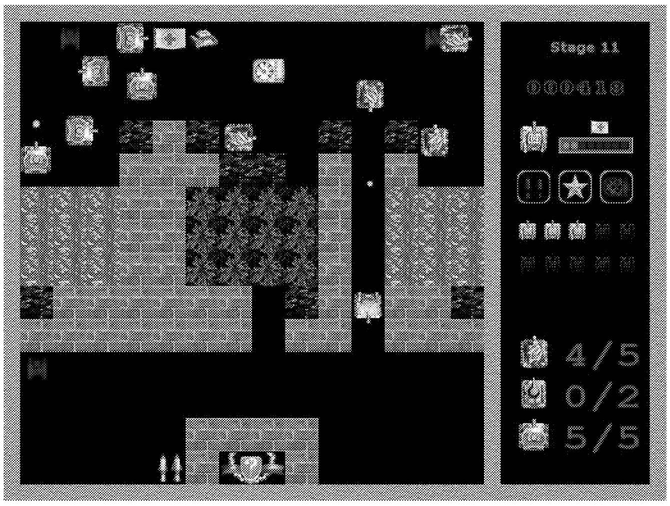


Figure 2: Screenshot of Awen Studio's Battleman

The requirements for this project included upgrading of the games engine to contain additions of new components and also the requirements on the artificial intelligence within the game play.

Battle City was chosen as there was a wealth of shareware applications of this game to use and these were observed during game play to gather requirements from and to use to compare with the finished product that this project produced.

Battle City was a project of adequate size with a variety of different requirement types to use the Volere Shells that was representative of requirements that occur in industry.

VOLERE SHELL RESULTS: REQUIREMENTS

The shell is in table format as it is easy to duplicate, read from and write into. There is also an excellent case to hold this information in a database with a simple data entry form to read and write with.

Table 1 shows our first requirement. Requirement FR1 describes that an NPC should be aware of a players presence and should respond to that. The requirement includes the rationale, source and measure as well as supporting materials and the importance of this requirement through its satisfaction arguments. This requirement contained no associated event, dependency, conflict or history so these fields are empty.

Table 1: Requirements Volere Shell (Clutton 2005).

Req. #	FR1	**Req. Type**	Functional	**Event**	
Description	An NPC shall respond to a player's presence, either firing or chasing.				
Rationale	This was a notable absentee in previous games, meaning the computer was easy to defeat.				
Source	Competition Analysis by Robert Clutton				
Fit Criterion	If in a fit condition, the NPC, upon recognising the player, should attack.				
Satisfaction	5	**Dissatisfaction**	5		
Dependencies				**Conflicts**	
Supporting Materials	Player Awareness in Requirements From Competition Analysis P1.4.2 Project Definition Document				
History					

A full set of Volere Shells for this case study can be found in (Clutton 2005).

AN EXTENSION: VOLERE TEST SHELL RESULTS

A Volere Shell is a requirements specification template. Through the lifecycle of Battle City it seemed a natural approach to extend the Volere Shell to incorporate test planning information. It was decided to have an associated test shell that extracted information from a Volere Shell. It would not be possible to include this information on the Shell itself, as the test script should include failure data which should be kept and repeated as necessary and not disregarded when the requirement is met.

In our example, the test scripts were run against the product and the shareware products used to gather the requirements. This data was then analysed to see if there was an improvement on the shareware versions available and to see if the projects objectives had been met.

Test scripts had been developed using the Volere Shell as a template. The Volere shell was used once again due the flexibility of the shell and the close traceability of requirements and testing. The following information has been compiled for the test scripts.

- Req. #: Requirement number from the requirements phase
- Req. Type: The type of requirement
- Fit Criterion: A measurement of the requirement
- Test Type: Black box, white box or play test
- Pass/Fail: A true or false value if the requirement has been met
- Fault Type: The type of error that occurred
- Error Description: Detailed description of the error
- History: Any previous failed tests associated.
- Inspector: The name of the person testing.

Table 2 shows the Test Shell for the requirement FR1. It follows the Volere Shell for the same requirements. Here we see a recap of the requirements' description and fit criterion. Then we add further information relevant for our tests. Our test type is observational as we want to be able to see this objective. There is a pass/fail mark and although this requirement has passed, it has not done so fully. An error description has been provided as to why this is so.

Table 2: Proposed Test Volere Shell (Clutton 2005)

Req. #	FR1	**Req. Type**	Functional
Description	An NPC shall respond to a player's presence, either firing or chasing.		
Fit Criterion	If in a fit condition, the NPC, upon recognising the player, should attack.		
Test Type	Observation		
Pass/Fail	Pass		
Error Description	Tank continuously attacks as there is only a small game world, however, the tank will escape if collision is about to happen.		
Inspector	Robert Clutton		

There is a strong link shown here between the requirement and the test, and with the format of the shells corresponding, it is easy to see this. Once again, these shells should be seen as complementary to any lifecycle processes, not exclusively part of any particular model.

A full set of the extended Volere Test Shells for this case study can be found in (Clutton 2005).

CONCLUSIONS

Effective software engineering methods are increasingly needed to support games development. To that end, this paper focused on the link between requirements and testing and how to support that process using Volere Shells.

The case study showed that this was workable in the context of a small-scale games development project, and that there appear to be a number of advantages in using Volere Shells. In particular they represent a flexible, lightweight, low-overhead approach to gathering requirements and deriving a consistent and thorough test plan.

The explicit linkage between requirements and testing has clear practical use – given the rapidity of requirements change that can be characteristic of games development projects, this linkage helps ensure that changes in requirements are fully reflected (which is especially useful for agile methods), and that no tests arsing (or made redundant) from changed requirements are inadvertly missed.

In the context of games development projects, the ability and freedom to adapt Volere Shells to fit the needs of each project is clearly useful. In addition Volere Shells can be used independently of the choice of software development process, whatever be it the unified process or some form of agile method such as XP.

Further work would involve a much more extensive case study, preferably examining the use of Volere Shells in the context of a full-scale commercial games development project.

ACKNOWLEDGMENTS

We would like to thank Dr Rob Saunders for his advice in the early stages of this work.

REFERENCES

Ashworth, C. & Goodland, M. 1989. SSADM: A Practical Approach. McGraw-Hill Publishing Co.

Beck, K. 2004. Extreme Programming Explained: Embracing Change. Addison Wesley.

Clutton, R., 2005. An Introduction of Artificial Intelligence into an Arcade Classic. Undergraduate Final Year Project, City University, London.

DeLoura, M. (Editor), 2000. Game Programming Gems. Charles River Media.

Eberly, D, 2004, 3D Game Engine Architecture, Morgan Kaufmann.

Gold, J. 2004, Object-Oriented Game Development, Addison Wesley, Essex.

Rees, E. & Fryer, L., 2003., IGDA Business Committee: Best Practices in Quality Assurance/Testing. International Games Developers Association.

Robertson, J. & Robertson, S. 2002. Volere: Requirements Specification Template, 9th Edition, Atlantic Systems Guild, London

Sommerville, I, 2001, 6th Edition, Software Engineering, Addison Wesley, Essex

BIOGRAPHY

ROBERT CLUTTON was born in Barking, England and went to City University, London where he studied for a Bachelors degree in Software Engineering. He graduated in 2005 and is currently working for British Telecom and has previously worked for Lloyds TSB, Group IT.

ANDREW TUSON was born in Dagenham, England and and was educated in Oxford and Edinburgh Universities where he studied Chemistry and then Artificial Intelligence. He has been research active since 1994 with over 25 published papers Dr Tuson joined City University, London in 1998 and is currently a Senior Lecturer in Computing.

Modelling and Prototyping for Psychological Time in Games

David England
Abdennour Rhalibi
School of Computing and Maths
Liverpool John Moores University
Byrom St
Liverpool L3 3AF,
United Kingdom
Email: d.england@livjm.ac.uk

KEYWORDS
Modelling, time, interaction

ABSTRACT

Time is an important factor in interaction and particularly so in real-time games. We described our approach to modelling user time in games in terms of temporal durations and temporal relations of user actions. We also present our notation PUAN, Pattern-Oriented User Notation for modelling temporal aspects of game play. Finally we present our Java Engine for Temporal Constraints, JETeC which is able to read PUAN descriptions to support prototyping, experimentation and testing of temporal aspects of game play.

INTRODUCTION

Time in computer games development is usually considered in terms of systems time. That is, developers usually think it terms of screen frame rates, CPU/GPU clock cycles used and network propagation rates. All these have an affect on the users' experience but represent a bottom-up approach to dealing with the temporal aspects of game play. In this paper we will look at time from the player's point of view or psychological time. We will consider how it can be modelled and how we can express temporal properties of games play that can be prototyped for evaluation and testing.

Time in Interaction

In standard approaches to human computer interaction, time is usually only considered in terms of simple response time to user input [Diaper 1989]. However, later researchers [Gray 1994] started to add temporal features to modelling the behaviour of users, in terms, not just of temporal durations, but also the temporal *relations* between the tasks that the user was carrying out. This latter feature is important when we are considering multi-tasking environments such as windows where users are switching between applications and are using applications in combination to perform a larger, overall task. Similarly in networked or multi-user environments, users will be trying to coordinate their activities with other users or agents in the network. Their performance will be affected both by the speed of the network and the rate or pace of response from other users.
These temporal factors at the interface have an effect on the temporal performance of the user, particularly on short-term memory when acting in a fast moving and complex environment. An often-quoted guideline in human computer interaction is that people can only hold 7 +/- 2 items in short-term memory at any particular time. However, this is only part of the story [Lindsay 1977]. Short-term memory as its name implies degrades quickly over a few seconds so any interruptions or distractions can wipeout its contents. Short-term memory is also subjected to ordering and frequency effects. So we are more likely to remember items that are presented at the beginning and end of a list of items. And with the interplay with long-term memory we are more likely to remember items that are presented more frequently. The term *items* does not simply mean a single digit or number. People can use chunking to put sets of numbers and digits together. Thus we can learn and remember phone numbers much longer that 7+2 items because we break them down into chunks of area code, local code and individual number.

The pace of events with which they have to deal also affects people's cognitive performance. There is some debate amongst psychologists as to whether people have an internal clock [Zakay 1990] against which they can measure the assign of time. Regardless of this people can feel driven by the pace of events at a computer interface and this can have a positive or detrimental effect on their performance depending on their level of ability.

Time in Game Play

The best games designers are probably intuitively aware of at least some of the above factors when they are designing interactive, real-time games. They know that the pace of a game can affect the player's successful progress through levels. They know how to move up the response times of enemies to give the game player less thinking time. They know that adding more enemies impacts on working memory and requires the user to learn new strategies for slicing their time between multiple enemies so that they can be defeated. They know that distracting elements at crucial times interferes with player performance. In general games designers will have some feeling for how these factors can be manipulated to present increasing challenges to players as they progress through the levels of a game. Psychological time then is one of the key components of good games design.

However, as software engineers, we are always thinking about how we can capture good design knowledge and decision making so that it can be re-used, tested and evaluated for a more planned and productive approach to games development. In the rest of the paper we will present our method for capturing and modelling these temporal aspects of design as patterns. We will then show how we can then use these pattern descriptions in a Java-based prototyping tool so they can be tested and evaluated.

PATTERN-ORIENTED USER NOTATION

Our Pattern-Oriented User Action Notation, or PUAN, is based on the User Action Notation of [Hartson and Gray 1991]. It has been developed over the years to cover more temporal aspects of interaction. We have applied it to cover interaction in Virtual Reality [England and Du 1998] and Multi-platform interaction [England and Du 2003]. It is more fully described in [Du 2004]. Here we will give a brief overview before giving some game-related examples.

The PUAN follows a similar template to the software engineering patterns. That is a pattern has a name, a description of the problem it is attempting to solve and the names of other patterns, which it uses, or references. In our pattern language, in addition to an informal problem description, we also have a formal PUAN description that can be executed by a prototyping tool. The PUAN formal description allows the description of temporal relations between tasks such as:

- Tasks are in sequence ,
- Tasks are truly concurrent &
- Tasks can be interleaved ||
- Tasks can be interrupted ->
- Tasks can be order independent <|>

The symbols above are used in the written description.

In addition we can query the start,
stop and duration time of a task and use temporal comparators to express temporal constraints. We can check whether tasks can start before, during or after each other and use conditionals as with any other programming language.

Example – Comanche 4

As an example let us present a moderately complex but partial scenario from a real-time simulation game, Comanche 4. We will take the Spetnaz rescue mission as a complex example (i.e. the author has not completed it). The overall task
at this level is to fly a helicopter to rescue forces from a train. The train is in a wooded area surrounded by enemy forces. The enemy consists of other helicopters, mobile SAM, tanks and armoured snowmobiles. The player has several sub tasks:

- Flying the helicopter
- Scanning the radar for enemies
- Visually scanning for enemies
- Choosing weapons
- Choosing targets
- Firing weapons

Additionally there are higher level tasks composed of these subtasks

- Achieve mission
- Plan a path of attack
- Find cover behind trees and terrain

Temporal constraints also exist in the scenario:

- Enemy vehicles will detect the player within a certain time and attack
- New enemy vehicles will join the scene if the mission is not completed within a certain time
- The mission has to be completed before too much damage is sustained.

We could proceed to further refine the scenario and describe it in a conventional engineering notation like UML but for our purposes we wish to examine the player issues. In the scenario we have several tasks going on at the same time with different temporal relationships between them. Some tasks are truly concurrent like flying the helicopter and scanning for enemies because we are using different input and output channels in terms of visual input to short term memory and motor control of the helicopter controls. Other tasks are interleaved because we are switching between different tasks such as scanning the radar and scanning the sky and ground for enemies. And some tasks have to be done in sequence such as choosing a target and firing a weapon. We can start to express these relationships more formally so we have a basis for testing and evaluating them in our prototyping tools. For example,

```
While (damage != 100%)
{Plan_of attack }
```

Plan_of_attack is our top-level pattern of interaction which could be applied to any number of games scenarios. In this instance it breaks down to

(Fly helicopter || (scan radar || scan scene) |
Choose weapon || (choose target, fire weapon))*

The asterisk is standard BNF notation for repeating tasks zero or more times. We have a collection of concurrent, interleaved and sequential tasks to accomplish our mission. We have written the Flying helicopter task as interleaved with the other tasks as there is a task dependency between the position of the player (helicopter) and choosing and firing at a target. There are further temporal dependencies as the enemies move, target the helicopter and fire. Additionally there is gap of opportunity between the firing rates of enemies.

Enemy Action
(move enemy || (target helicopter, fire)*)*

We can now express a relation between the player and enemy as

(Plan_of_attack | Enemy Action)

where the difficultly of the plan of attack is a function of the number of the enemy, N, as the more enemies we have the less time we have to carry out an attack task for each one. In temporal constraint terms

duration(Plan_of_attack) = F(N)

and

duration((scan radar || scan scene) | Choose weapon || (choose target, fire weapon) = F(1/N)

i.e. we have less time per enemy as their number increases. In terms of testing and evaluation we can play around with the value of N to find a balance between the level being too easy or hard.

As an aside there is a cognitive overhead in switching between the concurrent and interleaved tasks. The short-term memory for each task has to be re-established as we switch. In conventional interfaces we try and minimise the switching overhead with cues on the screen but in games it's a mechanism for adding to the challenge.

Looking at our collection we can see where the further challenge and level progression in the number of enemies facing us. This adds cognitive load in the scan radar and scan scene tasks, and subsequently the choose target task.

Prototyping the Scenario

Prototyping the scenario is achieved using our JETeC (Java Engine for Temporal Constraints) tool set to execute the PUAN description. JETeC provides an API which links the temporal descriptions in PUAN and to the scene objects written by the developer. JETeC creates a java thread for each task and uses the temporal constraint descriptions to manage the temporal relations and time checking of task execution.

As PUAN is a textual notation we can make changes to the temporal behaviour and properties of a game scene without changing the game artwork or other game objects. We can thus make hypotheses about the users' behaviour given certain temporal conditions and test whether those conditions hold.

There are some current limitations to our current toolset. The toolset does not enforce hard-real time. It can only give a notification that a temporal constraint has been broken. Secondly our tool set is itself a prototype or proof-of-concept vehicle to test our ideas about specifying and testing temporal properties. Hence it is not robust enough for development testing. However, we believe sufficiently demonstrates the value of specifying temporal constraints for games design and testing.

The JETeC Engine consists of two main components: the parser for reading PUAN descriptions, checking them and linking them with Java functions; and the scheduler. The scheduler is responsible for monitoring the behaviour of PUAN tasks which are converted to Java threads in the running prototype. The scheduler executes the PUAN description linking it will other components which provide the graphical interface to the game and the underlying logic of the game. It also monitors compliance for temporal constraints for the prototype as expressed in the PUAN. Thus it can take alternative actions if these are specified or flag up deviations from the PUAN when these are spotted. For example, a time duration may be exceeded or a sequencing of events or functions may fail.

FUTURE WORK

Our approach to temporal aspects of usability is just one approach to dealing with temporal issues. Other researchers have used Petri nets and other techniques from real-time systems development to tackle the problem. However these tools are often inaccessible to the designers of interactive systems. A simpler approach to temporal task dependency may be found in [El-Rhalibi 2005]. This paper looks forward to the new multi-processor consoles such as the Xbox 360 and Playstation 3, and proposes a method of partitioning games threads to make use of the new platforms. The same technique could be used to partition user tasks. In addition we can look at other work in the use of AI and scheduling to perform some a-priori analysis of PUAN descriptions to ensure they are workable and robust before we try to use them in prototyping. This will become an important aspect as games and PUAN descriptions become more complex, and thus harder for human developers to debug.

The new platforms also posed new challenges to games designers. As discussed at DiGRA 2005 developers have hardly begun to thing about the possibilities of using multi-threaded processing in games. Our approach to modelling multiple threads in games may support an approach to modelling games that take advantage of multi-processor platforms. At the same time it offers a way of testing and evaluating what are often difficult or even intractable problems found when developing multi-threaded applications.

Returning to our opening psychological theme there has been much work in user modelling as a means of testing interaction models. However, much of this work has not yet considered multi-tasking by users. More recent work by [Vera 2005] attempts to match a task description with a set of guidelines which describe optimum ways of doing tasks with task switching.

Overall then there is a great deal of potential from both the computer science theory side and the psychological modelling side in modelling complex scenarios involving time in interaction in games.

CONCLUSIONS

We began by considering the psychological aspects of time in interaction and games play. We can see that temporal issues are an important factor in designing for good games play. We gave an outline of our notation PUAN which aims to capture good design decisions of time in game play but allowing the modelling of patterns of temporal relations and durations. A toolset JETeC was briefly described which allows PUAN descriptions to be executed for testing and evaluation.

There is a great deal of scope for future work in terms of the psychological understanding of time in game play, the development of better models of time in computing and the development of better tools to support games researchers and developers, in dealing with the ever more complex and demanding development of modern and future games.

REFERENCES

D Diaper 1989, Task analysis for human-computer interaction, Chichester, West Sussex, England: New York: E. Horwood;

Min Du, 2004 “Temporal Patterns For Complex Interaction Design”, PhD thesis, Liverpool John Moores University

A El-Rhalibi, S Costa, D England 2005, Game engineering for a multiprocessor architecture, proceedings of DIGRA 2005, Vancouver, CA

England D, Gray P D, 1998 "Temporal aspects of interaction in shared virtual worlds", interacting with Computers, Vol. 11, pp 87-105,

England D, Du M, 2003 "Temporal Aspects of Multi-Platform Interaction" in Multiple User Interfaces: Multi-devices, Cross-platform and Context-awareness, Ahmed Seffah, Homa Javahery (eds) , John Wiley, 2003

E Gamma, R Helm, R Johnson, J Vlissides, G Booch – 1995, Design patterns: elements of reusable object-oriented software, Addison Wesley

P Gray, D England, S McGowan 1994, XUAN: enhancing UAN to capture temporal relationships among actions, Proceedings of the conference on People and computers IX, 1994

HR Hartson, PD Gray , 1991, Temporal Aspects of Tasks in the User Action Notation
PH Lindsay, DA Norman 1977, Human information processing: An introduction to psychology, New York: Academic Press

Vera, A., Howes , A., Lewis, R.L., Tollinger, I., Eng, K., Richardson, J. (2005). A Constraint-based Approach to Understanding the Composition of Skill. Invited paper: Cognitive Architectures in HCI (Ed. M.Byrne). HCI International , 22-27 th July 2005, Las Vegas, Nevada, USA

D Zakay, RA Block 1990 Cognitive models of psychological time, Erlbaum, Hillsdale, NJ,

BIOGRAPHIES

Dr David England is a principal lecturer in Computer Systems and head of the Computer Systems group at Liverpool John Moores University, School of Computing and Maths. His teaching and research interests are in Computer Graphics, Virtual Reality and Human Computer Interaction. More recently he has been involved with new media artists in producing experimental games that explore themes of Human Computer Interaction. These can be seen at www.hci-fun.org.uk

Abdennour Rhalibi is a principal lecturer in Computer Systems at Liverpool John Moores University, School of Computing and Maths. His teaching interests are in computer games development, AI and Animation. He was the founding programme leader of the Computer Games Technology degree course at Liverpool John Moores University. His research interests span AI and its applications to games and Computer Graphics. He is on the technical committee of many current games conferences and he the organiser of the Games Design and Technology Workshop at Liverpool John Moores University, now in its third year.

Space Syntax, Graph Theoretic Methods Applied to an Investigation into the Navigability of Large-scale Virtual Game Environments

Nicholas S.C. Dalton

University College London
Gower Street
London WC1E 6BT, UK

Email sdalton@cs.ucl.ac.uk

ABSTRACT
This paper suggests that the analytic techniques used in space syntax might play a role in the support of the design of certain types of games as they do of urban and building design. After briefly introducing the audience to the mathematics of the graph theoretic analytic techniques under discussion, the paper proceeds to present an analysis of one particular game map, Grand Theft Auto - San Andreas. In particular, the analysis is used to investigate the 'reward' structure of the game. The paper concludes that there is sufficient evidence to support the use of such techniques in the analysis of games: first, the syntactic structure of the game matches the intuitive layout of the virtual urban-structure, second, the placement of the game's 'rewards' are not placed randomly (an analysis of the reward-placement suggests possible superior placement-strategies based on the graph analyses) and finally, the intelligibility of the three cities in San Andreas appears to affect navigational ability, a finding which is substantiated up by a user-preference rating-poll for the cities.

Keywords
Game level design, navigation, intelligibility, space syntax.

INTRODUCTION
As In many ways, the difficulties of designing large-scale games, incorporating unrestricted movement (so called free roaming games), can be likened to the problems encountered in the design of complex buildings or urban landscapes. Games designers, like traditional urban planners, have the difficulty of crafting a landscape that will operate as anticipated. For example is one particular urban-layout more disorientating than another? Does one design favour one set of routes to those intended? Alternatively, might the design be so complex as to make it impossible for players to find their way without a detailed map? In large-scale games it becomes difficult to intuitively anticipate all possible a problems and solutions.

For the past thirty years, urban designers and architects have had similar problems and have looked to a analytic theory known as 'space syntax' to help interpret and measure the potential roles of space and pedestrian movement when making changes to the urban environment.

As games hardware continues to become more powerful, a number of game-environments have grown to the extent where they can be classified as massively large-scale[1]. This paper defines a large-scale environment as one that incorporates many route choices and typically requires many hours of exposure for users to become familiar with navigation in that environment. Games such as Grand Theft Auto (Rockstar Games. 2005) and Getaway (Team Soho -Sony Entertainment- 2003), typify the move to large-scale backdrops to support the narrative elements. In these cases, the design and construction of the environment becomes a considerable proportion of the design-budget. For games, the traditional process of build and test becomes problematic when seemingly inconsequential early-design decisions eventually lead to problems that only emerge during late end-user testing. One such potential problem-area is that of navigability - the ability of the average user to navigate successfully through an environment. It might be that a game is designed to be deliberately maze-like; alternatively it could be that an otherwise very good level is accidentally un-navigable. In these situations, user testing may show that the players are getting lost and perhaps becoming frustrated with the level design. Alternatively, the users, in confusing environments, may navigate unanticipated (by the designer) routes that by-pass some of the narrative elements of the game. In these circumstances, the solution may come from redesigning the level and in these cases, the size of environment and long construction processes may lead to long production delays.

This uncertainty of design outcome is reflected in traditional architecture and urban design. In this case, design oversights may equally be irreparable once the construction process has been committed. Over the last thirty years, traditional architecture and urban design has begun to apply a design theory known as 'Space Syntax' to help understand the 'movement variable' in advance of the construction process. In practice, space syntax helps

[1] A 'large-scale' virtual environment is defined as one that cannot be seen, in it's entirety, from a single vantage point. A new term is needed to define virtual worlds that are sufficiently complex that they can not only not be seen from a single vantage point, but that considerable time is required to navigate from one side of the world to another. This paper terms this a 'massively large-scale' environment, partly in reference to the category of games known as 'massively multi-user', although it should be stressed that not all massively large-scale environments are MMU (and vice versa).

the designer understand how design decisions might affect pedestrian flow and so avoid what is known as the 'empty windswept plaza' effect. By applying the analytical methods early on, architects and planners can to begin avoid early mistakes in the design process while they are still remediable.

This paper is premised on the likelihood that it may be possible to apply space syntax theories to the same kinds of design questions encountered in games as those found in large-scale architectural/urban design. The benefits may potentially go in both directions, while the design of games may learn from the theories and software tools used in building and urban design. It might also be that, changes necessary to facilitate level design might feed back to the real world context. Equally architectural theory suffers, like the natural sciences, from only being able to study the existing universe. Like natural scientists, it is hard for urban designers to perform an 'experiment' where two identical urban environments differ through a single variable under study. The use of game and virtual environments offers the potential for a truly experimental urban design science.

The first question, which must be addressed is, can gamers navigate in the same way that they do when moving though real space. The work of Ruddle, (Ruddle, R. A et al, 1997 , Ruddle, R. A et al, 1998) Conroy (Conroy, R.A. 2001, Conroy Dalton, R. 2003) and Haq(Haq, S. 2005) suggests that people in immersive environments certainly do navigate in a comparable way to movement patterns in real-environments. While further work needs to be done to reinforce these findings it seems reasonable that this work can be built upon by using this premise as a working assumption.

SPACE SYNTAX

The core work for the theory of space syntax is produced by Hillier and Hanson (Hillier, B. and Hanson, J. 1984,. Hillier B,1996.). While this theoretic field is constantly developing, the primary notions and techniques might be simplified down to this level; space syntax promotes the concept that 'space is the machine', that the logical unit of analysis for the interaction between society and the built form is the permeable space between buildings. Permeable space differs for different modes of transport. A canal becomes part of the permeable space for a boat or swimmer but part of the impermeable space for a driver or pedestrian. In London, Oxford Street constitutes parts of the pedestrian- and bus-permeability maps but not the car map[2]. Equally motorways/freeways and railways often form blockages to pedestrian movement.

Space syntax analysis frequently begins with the production of a permeability map for a 'spatial system' (building, village, town, city or neighbourhood). Initially, all spaces are identified. What constitutes a 'space' depends on the spatial-decomposition method chosen. For small buildings, spaces might simply be rooms. For larger urban systems, spaces are commonly defined as 'axial lines', these being the smallest set of the longest lines of sight that fully describe a system (Hillier, B. and Hanson, J. 1984).

Once the permeability-map has been decomposed into its discrete spatial units (typically axial lines) the units are processed to form a graph. The graph consists of a node representing each axial line. Where axial lines intersect, then an edge (a link) between both nodes is introduced in the graph. It should be noted that this is quite a different representation to the 'node-at-junctions' and 'edges-along-streets' representations that might initially be expected and is most familiar to transportation engineers. It should also be noted that these axial derived graphs are normally non-planar. Conceptually, an axial line can be seen as a unit of conceptual complexity that is to say that continuing in the same direction is simpler than remembering a number of twisting turns (Conroy, R.A. 2001,. Conroy Dalton, R. 2003)

There are a great number of operations that can be applied to any graph. Typically these are calculating global properties of a graph: its size, its girth, its diameter, whether the graph is closed, and if it is a small world graph, what is its chromatic polynomial. Space syntax differs by being interested in the local (the node or axial line) properties and how these relate to the global properties (the structure of the whole graph) of the graph. Core to this mechanism is the concept called integration.

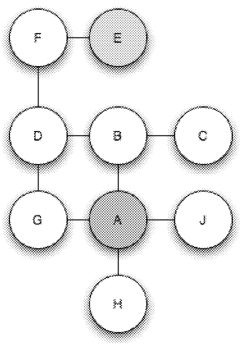


Figure 1 Example of a Simple Graph

To demonstrate the concept of integration, consider the simple graph presented in figure 1. This graph contains nodes labelled A and E. Beginning from node A, a minimum-spanning tree[3] can be constructed. The set of all nodes connected to A (set S_1) can be held to have depth 1; all the items directly connected to those items in set S_1 which are not in set S_1 (nor are starting-node A) are

[2] This is because private cars are denied access to this street, unlike pedestrians and buses. Therefore, the barriers to permeability may not always be physical but may be legislated.

[3] In space syntax terminology, this is called a justified graph or j-graph.

placed in Set S_2. This process continues, incrementing depths, until all nodes have been allocated to a set[4].

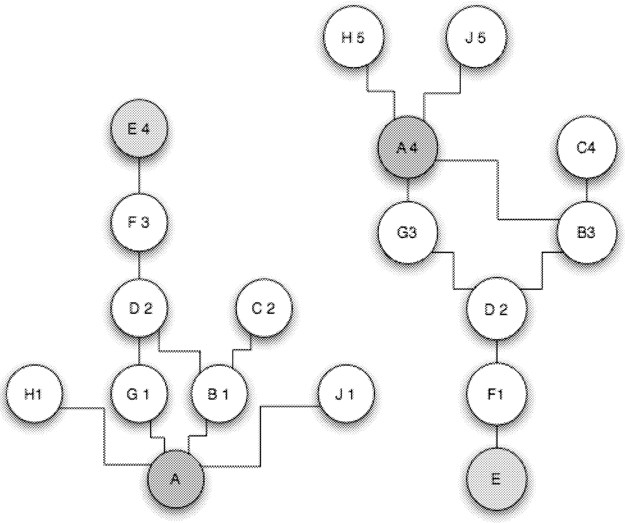


Figure 2 Distribution of depth from different origins

Each node has a depth attribute that is the length of shortest path length from that node to the starting node A (Figure 2). The total depth of the node A is the sum of all the shortest path lengths from A,(see Equation 1 Definition of Total Depth D for node i where d(j,i) is the smallest path length between node i and j in this example 15. Surprisingly, if this process is repeated using second node, E, as the starting point a different total depth value is produced, 27 in this case.

$$D_i = \sum_{j=1}^{j=N} d(j,i)$$

Equation 1 Definition of Total Depth D for node i where d(j,i) is the smallest path length between node i and j

In space syntax terminology, if a node has a relatively low total depth it is described as being 'integrated'. If a node possesses a large value (relative to the values of the other nodes in the system) then it is termed 'segregated'. Repeating this analysis for each node in the graph a spectrum of values is calculated, which are particular to the spatial system. However, by using the normalisation equations introduced by Hillier and Hanson (Hillier, B. and Hanson, J. 1984) values, known as the integration values, are calculated which permits comparisons between spatial systems of different sizes.

The above computations on the graph, formed from the intersection of the axial lines, can be repeated for different systems. Visualising these 'integration' values by colouring the original axial map, it is found that busy streets (such as London's Oxford Street) are the most integrated. Cut-off locations (back alleys, cul-de-sacs) tend to have segregated (low) integration values. These values can be visualised by plotting them on the axial map as a spectrum of colours from red (highly integrated) through orange, yellow, green, light blue, dark blue (for highly segregated). For real urban locations it is found that the integration values reflect something of our knowledge of configuration of the streets. Busy streets tend to be towards the integrated (red) end of the spectrum and quite 'dead' streets tend to be towards the segregated (blue) end. Previous research has shown that if the pattern of pedestrian movement is noted (by counting the number of people walking past a notional 'gate') and compared to the integration values of the axial lines, that pedestrian flow values tend to strongly correlate with the integration values (Hillier, B., Penn, A., Hanson, J., Grajewski, T. and Xu, J. 1993).

[4] The starting node can be considered to be at depth zero and therefore in a set of its own, S0).

Note that in this form of analysis the only input to the 'model' is the configuration[5] of space. This analysis does not include attractors, (shops, offices, restaurants) or generators (train, tube bus stations, parking lots). Neither does the analysis include land use patterns, road quality or the 'mental models' of average pedestrians. This minimal input modelling makes syntactical analysis very suitable for early-stage conceptual design.

Finally, in large-scale analyses, a parameter called 'radius' is introduced. This creates an artificial window centred on the axial line under study. With a Radius of r the total depth of a node, j, is limited to those whose depth-values are less than or equal to r. The use of radius r= 3 has been broadly found to be a suitable one for use in pedestrian analysis and is, in the absence of any observational data, the radius used in this paper.

ANALYSIS OF THE GRAND THEFT AUTO ENVIRONMENT

This paper is a part of a larger investigation into navigation in large-scale virtual environments. The focus of this work lies in the types of environment found in desktop VR and three-dimensional game worlds. This paper will concentrate on the study of one game, Grand Theft Auto - San Andreas (GTA-SA) (Rockstar Games. 2005), an example of the current generation incarnation in a series of 'free-roaming games'. The foundation to this analysis is the similarity between the game environment and real-world urban environments. That is, the theatrical and narrative elements of the game are dependant upon the users' mobility such that they are generally free to move from one location to another using any navigational method they may chose (as per real life). The game Grand Theft Auto is set in a notional universe/state that is intentionally similar to California. There are three fictional cities, which bear comparison to San Francisco ('San Fierro', centre left on Figure 3), Los Angeles ('Los Santos', bottom right on the map) and Las Vegas ('Las Venturas', top right on the map). The cities are connected

[5] A configuration is defined as a set of spaces and spatial relations where each space affects and is affected by all others; a change to one space will ultimately produce a change to the status of all other spaces.

by a series of highways. The global syntactical model[6] is presented in Figure 3. It can be see from this analysis that the most integrated point (coloured red-orange) is the central highway linking the cities. This would suggest that if observational studies could be conducted (as per a real-world study) that the highways shown in red would tend to be those most visited. This is not a surprising finding; in real US cities the interstate/motorways frequently have the highest traffic flow counts.

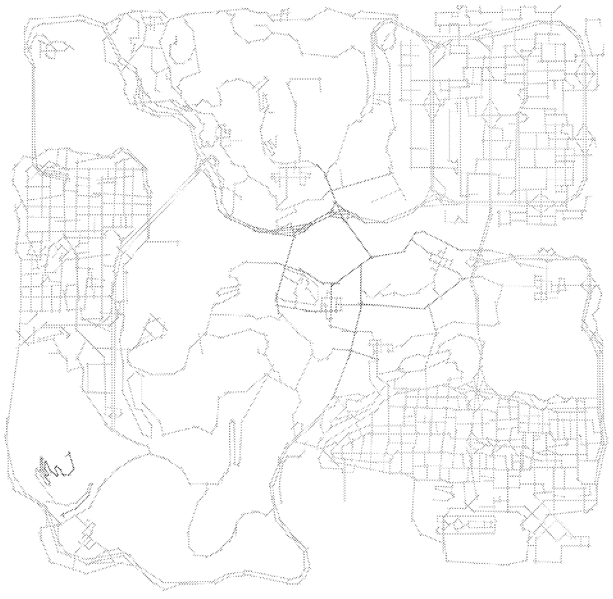


Figure 3 Global Pattern of Movement in the Game

In practice the narrative thread prevents global free movement before a number of missions are completed. As a consequence it makes sense to investigate each city as a separate entity.

If we look at the integration pattern, such as that in Figure 4, of Los Santos we find that the game's use of space is consistent with its configurational layout (as analysed using space syntax methods). For example, the downtown area is centred on the most integrated (red) areas. The docks are located in highly segregated (blue) locations are as the high status buildings (the blue areas to the north/top). These patterns are typical of those found in real cities.

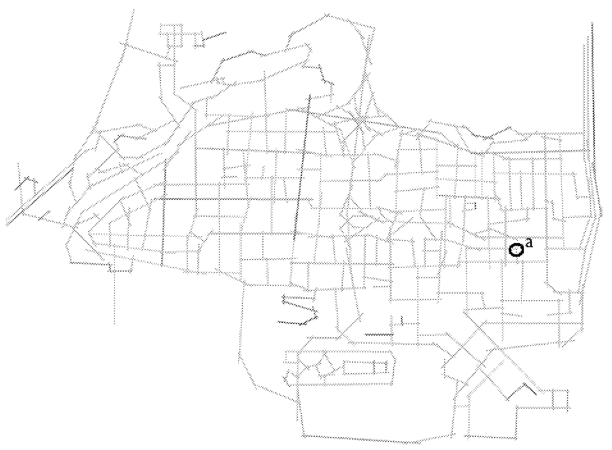


Figure 4 Global Integration Map of Los Santos. Game's Home-base Labelled 'a'

THE PLACEMENT OF 'REWARDS'

Is there a way of testing the applicability of space syntax to the design of large-scale games like Grand Theft Auto? As a preliminary to further, proposed research into patterns of user-navigation, there are a number of simpler studies investigating the design of the game (rather than its use) that may be investigated. One of the aspects of GTA-SA is the existence of 'odd jobs'. The player is presented with a number of 'odd jobs' such as overwriting 'tags[7]' (graffiti placed by the games designers); overwriting tags (by replacing it with your own) gains 'respect' points and helps the player's overall objectives. According to the official strategy guide (Bogenn, T and Barba, R. 2005), if all the tags are overwritten then special rewards become available. Each tag can be removed in any order and non completion does not effect the narrative or completion of the game. Each city, within the game, has a number of similar odd-job activities that are available in this way: photographing key views in San Fierro and finding 'lucky horse shoes' in Las Venturas.

Clearly, a game like GRT-SA is a game designed to be explored. The presence of location-based rewards is indicative of this, as the rewards cannot be collected without extensive exploration. The challenge of these rewards lies in the difficulty of finding them. Note that the game dynamic is quite clear here, if the items are in locations that are too segregated then the players is unlikely to chance upon them. If all the tag locations, for example, were in central and frequently visited (integrated) locations then they would be too simple to find and the challenge and skill to finding them would be diminished. Obviously, the game must strike a balance between placing items in obvious locations to encourage the player to begin collecting reward points and locating some rewards in the most remote places in order to create rarity (and so add value to the difficulty of finding them).

[6] Rockstar Games published the original maps, upon which the axial map is derived.

[7] In graffiti culture "tagging" is a term that describes a graffiti artist's personal logo, which is incorporated into their graffiti art.

From an architectural point of view, the question becomes one of whether the locations of the tags are a simple random selection from all spaces or does their placement reflect an understanding of the underlying spatial structure of the game world? Furthermore, we need to ask if the integration values, according to space syntax analyses permit the exposure of this implicit knowledge and aid the understanding of the configurational structure of the game space.

One preliminary test is to examine the distribution of integration values for the location of game 'gifts' and more specifically 'tags'. This paper makes the basic assumption that each tag is related to its nearest driveable street and hence may be assigned an integration value matching this street. The distribution of the integration values of all the streets may then be analyzed. Taking the subset of all streets, which have are associated with a tag, (S_t), it is possible to restate the above question in a statistical form. Is the distribution of the sample set, S_t, a random sample drawn from the general distribution of integration values (S) or not?

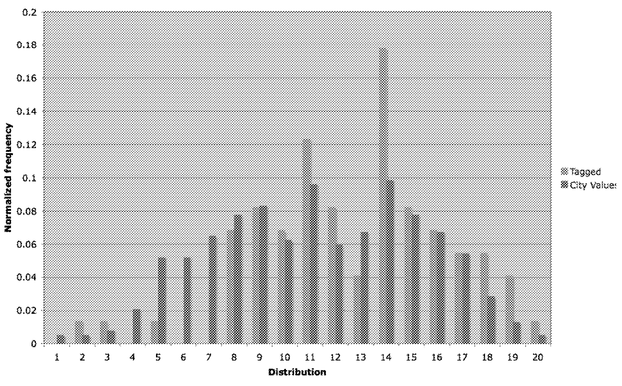


Figure 5 Normalised Frequency Distribution of the Global Integration Values of 'Tags' in Los Santos

This question can be partially visualized by creating the histogram in Figure 5. To permit the comparison between the different sample sizes the values have been normalised. By inspecting this chart, it can be seen that the tagged axial lines are disproportionately skewed. At the right-hand side of the histogram the number of tags is higher than would be expected, if the tags followed the distribution of all lines. Namely, there are more tags on highly integrated and hence highly visited lines. This follows the paper's initial hypothesis that a large number of tags should be places on highly visited lines in order to encourage the player to start their 'tag' collection. The high numbers of mid-range values (bin 14 on Figure 5) are found in the spaces near the starting point (or home base) of the game. This reflects the natural assumption that the tags are identifying 'gang turf', that is that they are centred on the opposing gang's territory.

Here the conflict between a good game and a good simulation is evident. It would be expected that a good simulation would focus 90% or more of the tags in gang territory and close to the geometric centre of their region, whereas the greater urban distribution of tags supports the 'treasure hunt' aspect of the game. Finally, it can be seen (on Figure 5) that the segregated bins (4,5,6 and 7) are mostly empty and that the most segregated bins (2 and 3) have a higher than expected number of tags. Bins 1-3 represent the most segregated (blue in Figure 4) locations. This is where the player has to be most persistent and diligent in the collection of tags (or where they must cheat and use a guide-map of tag-locations). Why are the bins 4-7 mostly empty? By checking the locations of these axial lines, it can be seen that these spaces tend to be the ones leading to the most segregated lines rather than the most segregated. Why place the reward halfway down a long, winding track when it can be placed at the end of it?

Such a visual inspection of the histogram can be supported by performing a Z-test between the set of 'all axial lines' (the population) and the sub-set of 'tagged axial lines' (S_t). The outcome of the Z-test produces a value of 7.66, which is larger than 1.96 suggesting that, we can say with a confidence of 95%, that the sample sub-set of tagged axial lines (S_t) is not a random sample of the population.

If this test is repeated using the integration radius three (r3) values (rather than integration values, as per the previous example), the Z value changes to 10.97. This result suggests that the positioning of the tags relate more strongly to the local centres rather than to global spatial structures (often the vehicular as opposed to pedestrian movement patterns). This result should be taken with some reservation, however, as the map analysed is the virtual 'vehicular map'. Were a detailed pedestrian permeability map available then a more subtle use of space might begin to emerge.

This experiment can be repeated by examining the 'snapshot' locations of San Fierro. In this city, the player has to take photographs of specific key locations. In this case, we might expect the photograph points chosen to be well distributed and if previous experience is any guide skewed towards more integrated (well-used) locations. The Z-test value for this is 5.54; again this gives us more than a 95% confidence that these points are not randomly selected from all available streets. For radius 3 (local or pedestrian movement) a Z value of 42.20 This is reinforces the findings from Los Santos that better results might be found by using a more detailed pedestrian-level map.

Finally, locations of the lucky 'horseshoes' in Las Venturas can be examined. In this section, the greater the number of lucky horseshoes that a player finds, the greater their luck on the gambling tables. Repeating the Z-test for Las Venturas on integration values, a Z value of 5.59 is found. Interestingly, this is very similar to the value for Z for global integration in San Fierro. Perhaps this suggests that further investigations might demonstrate that the choice of hiding places in San Fierro is similar to that for Las Venturas. The radius three (pedestrian scale) Z-test value is 9.95; again this reinforces the suspicion that analysis with a pedestrian permeability map would produce more clear-cut results.

INTELLIGIBILITY, NAVIGATION AND USER-PREFERENCE

In the field of space syntax, one factor that has been suggested is an emergent property of a city or urban region is the property of 'intelligibility' (Hillier B 1996) Intelligibility is the generally defined as the strength of the relationship between a system's local and global properties. Experientially this is a formal measurement of 'the degree to which, how much of what I can see, can guide me in my global movement through the whole system'. An alternative description of intelligibility might be how 'maze-like' is the system? One way of measuring intelligibility is to examine the correlation coefficient (r-squared) between a global measure (like integration) and a local one (such as integration of radius 3 or connectivity). In the real world low intelligiblity factors are common in places like desolate houseing estates/housing projects.

If the three cities are considered isolation, it can be seen that Los Santos has an r3/integration r-squared value of 0.648, San Fierro of 0.807 and Las Venturas of 0.496. It would be interpreted that San Fierro is the most intelligible and least maze-like of the three fictional cities and Las Venturas the least intelligible.

One independent discussion board (Neoseeker 2005) asked GTA-SA players the question "What is your favourite City?" Respondents were able to provide answers in any format they cared; by reformulating the results into a score for each city[8]. It was found that Los Santos was the most popular city (with a score of 92), followed by San Fierro (76) and finally Las Venturas (score 48). There were 53 clear (and encodable) respondents to this particular question. One hypothesis is that Las Venturas is the last city to be visited in the narrative and so has less exposure than the others. However, an alternative hypothesis, and the one which is suggested here, would be that an unintelligible (more maze-like) city would be less amenable to free-roaming exploration than a more intelligible one. Intelligibility factors of 0.807 (San Fierro) and 0.648 (Los Santos) are fairly high while the value of 0.496 (Las Venturas) is relatively low. While the relationship between intelligibility and user-preference is clearly not causal (a highly intelligible city is not necessarily going to make it preferable) it could be argued that it is at least facilitative (i.e. easier to make a 'good city' if it has an intelligible layout) of ease of navigation and hence good game-play.

[8] The answers were re-coded into a numerical score. Where a clear preference was given for one city, a score of 3 was assigned. If an unequivocal *dislike* for a city was expressed, it was ranked 1 of 3, with a score of 2 being reserved for the middle category: neither a marked preference nor dislike expressed. If a respondent simply did not mention a city, no score was given. The scores were summed for all answers.

CONCLUSION

First, it has been demonstrated that an analysis of the hidden 'rewards' for different cities, consistently produces a non-random shift in the distribution of the integration values of the locations of these rewards. Furthermore, the data indicates that these 'rewards' are even less randomly distributed if the player is searching on foot as opposed to driving. This result is consistent with the logic that the game was intentionally designed to create a simulated experience of urban space. Secondly, these results are consistent with this paper's hypothesis that the space syntax methods presented here are applicable to the analysis of game environments as well as traditional architecture and urban design.

Finally it has tentatively been shown that the intelligibility of a city may be consistent with the players' subjective impressions of the cities. Again, this contributes to the hypothesis that the design factors applicable for real cities are relevant to the virtual/game domain.

The work presented here is the initial stage of a larger investigation into the movement patterns of players in immersive games. The hypothesis that movement patterns in immersive games would be similar to those found in the real world will be better supported by a detailed observation of the movement patterns of players in the environment. The navigational patterns can then be compared to and correlated with the patterns of integration produced by space syntax analysis. It is intended that this be applied to a large number of sample worlds.

Clearly, more work needs to be done before it can be established that there is a reliable link between world design and player experience as mediated through space. The design of massively large-scale games like GTA-SA creates a number of novel demands on current spatial analysis methods. For example, is it possible to examine the role of narrative in the game and the effects this has on the learning of the urban layout? What is the relationship between spatial configuration and narrative structure? These kinds of questions will necessarily demand changes to and new insights into current real-world space syntax theories.

ACKNOWLEDGMENTS

My thanks to Ruth C Dalton for her assistance in proofreading and formatting.

REFERENCES

Bogenn, T and Barba, R. 2005, *Grand Theft Auto San Andreas Official Strategy Guide*. Pearson Educational. 2005.

Conroy, R.A. 2001, *Spatial Navigation in Immersive Virtual Environments*, PhD Thesis, submitted to the University of London, 2001.

Conroy Dalton, R. 2003, The secret is to follow your nose: Route path selection and angularity, *Environment and Behavior,* 35, 2003, 107-131.

Haq, S. 2005, Comparison of Configurational, Wayfinding and Cognitive Correlates in Real and Virtual Settings. In *Proceedings of 5th International Space Syntax Symposium*. Delft, 2005.

Hillier, B., Penn, A., Hanson, J., Grajewski, T. and Xu, J. 1993, *Natural Movement: or, Configuration and Attraction in Urban Pedestrian Movement.* Environment and Planning B, 1993.

Hillier, B. and Hanson, J. 1984, *The Social Logic of Space*. Cambridge University Press, Cambridge, 1984.

Hillier B 1996. *Space is the Machine*. 1996, London, Cambridge University Press, 1996.

Neoseeker 2005, http://www.neoseeker.com/forums/index.php?fn=view_thread&t=468284. Accessed 2005.

Penn, A. 2003, Space syntax and spatial cognition: Or, why the axial line? *Environment and Behavior,* 35, 2003, 30-65.

Rockstar Games. 2005, Grand Theft Auto – San Andreas. London, 2005.

Ruddle, R. A., Payne, S. J. and Jones, D. M. 1997, *Navigating Buildings in "Desk-Top" Virtual Environments: Experimental Investigations Using Extended Navigational Experience.* Journal of Experimental Psychology: Applied. 3(2), 143-159, 1997.

Ruddle, R. A., Payne, S. J. and Jones, D. M 1998. *Navigating Large-Scale "DeskTop" Virtual Buildings: Effects of Orientation Aids and Familiarity.* Presence: Teleoperators & Virtual Environments, 7(2), 1998, 179-192.

Team Soho (Sony Entertainment) 2003, Getaway, 2003.

BIBLIOGRAPHIE

NICK SHEEP CONROY DALTON born in 1963 Studied Appropriate Technology at the University of Warwick (UK). He worked as a computer science researcher at UCL and went on to engage in research in to 'Space Syntax'. In 1998 he setup the Masters In Virtual Environments in the Bartlett School of Graduate Studies. Currently he has returned to pure research as a EngD PHD student.

ONLINE GAMES RESEARCH

Requirements for Communication Frameworks for Mobile Games on Ad Hoc Networks

Stefan Fiedler
Departement of Media Informatics
University of Ulm
e-mail: stefan.fiedler@uni-ulm.de

Michael Weber
Departement of Media Informatics
University of Ulm
e-mail: weber@uni-ulm.de

KEYWORDS
Massive Multiplayer Games, Mobile Games, Context Awareness, Ad Hoc Networks

ABSTRACT

Mobile computer games are not only under development by the industry but also an area for researchers to explore new genres for mobile environments. The new generation of mobile game consoles allows new types of applications by supporting wireless networks. In this paper we introduce an approach that enables mobile games to communicate using highly dynamic mobile ad hoc networks. In order to allow for a fluent game play, we classify the interactions provided by a game according to their network requirements into different communication categories. We present a framework that incorporates these requirements and provides context information for the game. Then the game is able to use this context information to integrate it into the game logic.

Introduction

Multiplayer support is nowadays an essential feature for games to be successful. Especially Massive Multiplayer Games, where several thousand players interact in the same virtual world over the internet enjoy great popularity. With the large number of participants new challenges regarding consistency, persistence and synchronization arise.
With the increasing spread of mobile devices computer games become part of their basic configuration. But most computer games on mobile phones, PDAs, and early mobile game consoles are limited to simple single player games or multiplayer games with very few participants. The new generation of portable game consoles offers new opportunities, based on their superior hardware. The most important feature is their networking capability: With the introduction of wireless networks, especially Wireless LAN, ad hoc networks can be formed.
Nowadays games on these new mobile consoles support only a limited number of local players. All Players must stay in direct radio range and thus the same broadcast area. Such wireless networks provide quite the same conditions that were existent in classical Local Area Networks. The new opportunities provided by the new consoles are left unused, i.e. ad hoc networks and the mobility of players.

Related Work

Some games available are already designed for mobile devices and take advantage of their additional features. Games designed for mobile scenarios still under research are Mixed reality and Pervasive games which try to incorporate the real world context in which the players reside into the story and even in the handling of the games [BML05]. These games often use localisation mechanisms to provide this context.
Nearly all Massive Multiplayer Games in the internet use a client server communication architecture [Kus05]. The main reason for this is the simple control over the system. Persistence is easily accomplished, and synchronization algorithms are only needed between servers with a known infrastructure. Consistency is not a problem either as there is only one defined game state at the server cluster.
In order to operate Mobile Massive Multiplayer Games without central infrastructure, clients need to establish ad hoc networks to communicate with other players. But these networks introduce new problems like node failure and network partitioning. With nodes only being able to communicate within their direct neighbourhood only parts of the game state are accessible. Similar problems can be found in the areas of distributed file systems and mobile databases. [MS04]. In the disconnected state the clients usually work on a locally cached or replicated data set and propagate their updates to the server when they are online again. An essential difference to the scenario described here is the fact that they synchronize with a central server or database that updates the common global state and resolves inconsistencies.

Requirements

The goal of this paper is to present a framework that enables Massive Multiplayer Games on mobile gaming devices without a central infrastructure. All communication relies solely on peer to peer connections over multihop ad hoc networks. This enables nodes to communicate indirectly through additional nodes in between them, even if they are not in direct radio range.

Through the limited radio range of these consoles and their constant mobility we must assume that the ad hoc networks are built of dynamically changing subnets. There may occur network partitioning and fusion when nodes are regrouping. Especially in the case of network fusions a common state has to be found which is acceptable for all participants, while the game is running continuously and seamlessly onward. Furthermore the connection quality between the communicating nodes within a partition depends heavily on the network topology and has enormous impact on the game and thus has to be taken into account by the game logic itself.

The different forms of interaction in games [EPB05] result in different requirements referring to latency, persistence, and synchronization. For instance one class focuses on reactivity, which is especially important when interacting directly with other players. Other forms of interaction require a reliable transaction mechanism, for example when trading items with other players. Another class is indirect unit commands used for example in real time strategy games, which tolerate a moderate latency and the nodes are typically kept strictly synchronized.

Framework

A Framework for games that take these requirements into account must provide a content based communication interface. Depending on the requirements of every object their desired connection properties must be specified individually. To get to a profound model we propose a classification of typical interactions found in computer games taking in account communication parameters like network properties, consistency, and persistence.

Network Properties

The effect of latency and jitter on games for example depends heavily on the genre and has great impact on the subjective perception of the game flow. Jitter should be kept as small as possible and uniform especially in direct player interaction.

Direct User Interaction A latency of 100-225 ms enables direct interaction between players as is needed for e.g. fights [Hen01]. Higher latencies break the feeling of direct interaction and can lead to unfair game results.

Indirect Unit Commands Up to 500 ms are acceptable when using indirect commands to control characters, comparable to those used in real time strategy games [BT01].

Message Transference Even larger latencies are tolerable for the third category that includes for example messaging or item trading. Durations of several minutes up to hours per transaction are acceptable.

Consistency

As stated earlier not all events require strict synchronization to keep the virtual world consistent [LKD$^+$04]. For example no strict synchronization is needed if the character positions are transmitted in absolute coordinates allowing the game even to compensate packet losses for a certain amount of time by extrapolating the characters position. Since every node can communicate only with its neighbourhood new weak synchronization mechanisms must be provided to compensate the lack of a global view of the game state.

Persistence

In pure peer to peer architectures persistent storage and availability of objects is a major challenge. This is especially true for ad hoc networks in which the place where to store the object has to be chosen carefully. In order to guarantee high availability techniques similar to peer to peer approaches used in file sharing networks can be used.

Character Properties We propose to store all character properties and items associated with it on the player's device. Whenever a participant leaves the network no further interaction with that player or its associated items is necessary or possible.

Environmental Objects Objects not associated directly with a player could be handled by different methods. Many properties and objects in games have a volatile character and are only managed during runtime and are not stored permanently. If objects are persistent the framework has to replicate them in an appropriate way. These replicates don't have to represent exactly the same state. This is often not even possible due to network partitioning.

Messages Another category of persistent objects are messages or transactions between players. The nodes have to store and forward them in a reliable way to the receiver. To achieve this even in paritioned networks where users are not allways in reach, network context information can be used, for example if the framework saves a buddy list of players it has often contact to [DFL01].

Game Context

Users should be aware of the games context like the quality of network conditions or currently available players [Hen03].

Adaptation of Communication In order to map the different interaction classes to the game, it must be aware of context state of its surrounding. The framework collects this information and provides it for the game. This enables the game to estimate which environmental objects can be perceived by other players and thus determine which objects have to be synchronized. By taking into account the physical position of players a natural subdivision of the virtual world evolves. The division of the world in smaller parts allows a more efficient synchronization. A further optimization is to synchronize only objects within the player's sight. Weaker mechanisms can be used to synchronize objects not in direct visibility or far away from the player.

Integration into Game Logic These context information are not only used internally by the game to adapt the communication. They are also presented to the user when appropriate. This may vary from simple technical presentation in the user interface to direct integration into the logic of the game like instancing techniques used in massive multiplayer games.
In order to evaluate the practicability of such a system we are developing a prototype emulating such a game using a network simulator. We chose the network simulator Jist/Swans that allows an efficient simulation of network traffic with many nodes over a long period of time [Bar04].

Conclusion

We presented essential aspects that must be considered when implementing massive multiplayer games for mobile consoles without a central infrastructure. A major issue is the classification of interaction based on network requirements. This enables fluent gameplay despite largely varying connection quality in a dynamic multihop ad hoc network. We pointed out that integration of environmental context allows the game to incorporate technical conditions of the network and social context information on the one hand. On the other hand it is able to integrate context information into the game logic itself, and to present this information to the user. Considering these aspects a framework is developed that provides an interface through which an application can communicate and receive context information. In our future work we intend to further extend our game logic covering all essential communication classes to test our framework with network simulations. Using these network simulations we will evaluate and verify different algorithms and parameters on synchronization, persistence and context information.

REFERENCES

[Bar04] Rimon Barr. *An efficient, unifying approach to simulation using virtual machines.* PhD thesis, Cornell University, May 2004.

[BML05] Steve Benford, Carsten Magerkurth, and Peter Ljungstrand. Bridging the physical and digital in pervasive gaming. *Commun. ACM*, 48(3):54–57, 2005.

[BT01] P. Bettner and M. Terrano. 1500 archers on a 28.8: Network programming in age of empires and beyond. In *Game Developers Conference 2001, San Jose, CA*, March 2001.

[DFL01] James A. Davis, Andrew H. Fagg, and Brian N. Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. In *ISWC '01: Proceedings of the 5th IEEE International Symposium on Wearable Computers*, page 141, Washington, DC, USA, 2001. IEEE Computer Society.

[EPB05] Daniel Eriksson, Johan Peitz, and Staffan Bjrk. Enhancing board games with electronics. In *PerGames '05: Proceedings of the ACM Pervasive workshop on Pervasive Games.* ACM Press, 2005.

[Hen01] Tristan Henderson. Latency and user behaviour on a multiplayer game server. In *NGC '01: Proceedings of the Third International COST264 Workshop on Networked Group Communication*, pages 1–13. Springer-Verlag, 2001.

[Hen03] Tristan Nicholas Hoang Henderson. *The effects of relative delay in networked games.* PhD thesis, University of London, February 2003.

[Kus05] David Kushner. Engineering everquest. *IEEE Spectrum*, 42(7):28–33, July 2005.

[LKD+04] H. Lu, B. Knutsson, M. Delap, J. Fiore, and B. Wu. The design of synchronization mechanisms for peer-to-peer massively multiplayer games. Technical report, Penn CIS, 2004. Available online: `http://www.cis.upenn.edu/~hhl/Papers/MS-CIS-04-xy.pdf`, last accessed 20.7.2005.

[MS04] Bela Mutschler and Günther Specht. *Mobile Datenbanksysteme.* Springer, 2004.

Optimization of Multi-Player Online Game Server based on Predicted Dynamic System

Soon-Jeong Ahn , Woo-Suk Ju
Ying Quan , Choong-Jae Im
Dongseo University, Busan, South Korea
E-mail : {sjahn,savrang}@dongseo.ac.kr,
emilyquan@hanmail.net, dooly@dongseo.ac.kr

Keywords :
Online game server,Prediction based dynamic thread pool system

Abstract

Online game servers usually has been using the static thread pool system. But this system is not fit for huge online game server because the overhead is always up-and-down. Therefore, in this paper, we suggest the new algorithm for huge online game server. This algorithm is based on the prediction based dynamic thread pool system. But it was developed for web ser vers and every o.1 seconds the system prediction the needed numbers of threa ds and determine the thread pool size. Some experimental results show that the check time of 0.4 seconds is the best one for online game server and if the number of worker threads do not excess or lack to the given threshold then we do not predict and keep the current state. Otherwise we apply the prediction algorithm and change the number of threads. Some experimental results shows that this proposed algorithm reduce the overhead massively and make the performance of huge online game server improved in comparison to the static thread pool system.

1 Introduction

Computer online game is a kind of game that includes hundreds of players distributed over WAN scale networks. Now, there have been two fundamental server architectures proposed Peer-to-Peer and Client-Server (C/S) for communicating each other. We concentrate on the research based on C/S model. To construct C/S online game, we need make a server program to do with players' requests and send related data to specific clients. For designing this program we need first know some characteristics about online game. First, every player requests game packets in the time of o.1seconds constantly until the game is ended and its processing time must be very short. Second, a huge of requests is transferred to game server from hundreds of player simultaneity. Third, requests need to be response and the response packet are sent to related clients by game server in time but its size is extremely small in contrast with web server. Finally, normally, the number of player is below 1000 in the same game server and take the static thread pool system.
In this paper, we experiment with below 1000 users based on prediction based dynamic thread pool system. But we find that the system was not fit for huge online game server because overhead is

up-and-down constantly and the addition and reduction process happened frequently. So the the amount of processed packets is reduced. Therefore, we propose to set the threshold for the number of working thread and find the fact that the check time must be increased by some experiment. Under the se condition, we apply the prediction-based dynamic thread pool system to huge online game server. Some experimental results show that our proposed algorithm can make the overhead reduced and the performance of online game ser ver increased. This paper is composed as followings. We first introduce some characteristics of current online gam e server and the theories of static/predic tion based dynamic thread pool system in Section 2. We suggest the our proposed algorithm for huge online game server in Section 3. Finally, we compare the performance of the proposed online game server with static thread pool system and give some results in Section 4. Conclusions are presented in Section 5.

2 Previous Works

The client's requests transferred by the TCP/UDP layer are processed in worker thread . Fig 1 shows the overall architecture of thread pool system.

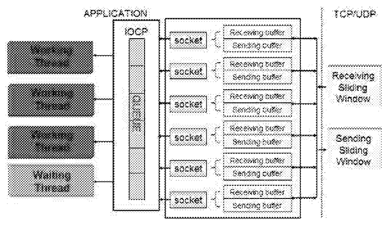


Fig. 1. architecture of thread pool system

There are two methods to determine the number of worker threads, called static thread pool system and dynamic tread pool system. In this section, we will introduce the static thread pool system for online game server and predi ction-based dynamic thread pool system for online game server.

2.1 Static thread pool system for online game server

This system use the fixed worker thread pool size as follows .

$$\text{worker thread pool size} \\ = \text{the number of CPU} \times 2$$

This means that the size of thread pool should be two times of the number of CPUs on the host machine [6]. Normally, it is considered as a start point and then is tuned according as the ability of processing packets and overhead to game system in order to further optimize the size of thread pool. Because there is not any time-waste on creating and destroying threads during the runtime, it reduces the response time. And it also avoids consuming the resource of server such as CPU usage and memory, when it creates overfull threads in game server. However, from the fixed the pool size, it can not use the system resource effectively, thus it can make lower the game server performance lower [2]. To solve this problem, it is desirable to have a kind of thread pool which can configure itself based on the current status of thread pool. That is dynamic thread pool system. D. Xu and B. Bode [7] suggested the dynamic thread pool system using the heuristic algorithm. J.H. Jung et al. [3] suggested that the worker thread pool size is determined by the prediction-based dynamic thread pool system.

2.2 Prediction-based Dynamic Thread Pool System

It is a combination of a worker thread pool and a pool manager thread. Pool manager thread can tune the size of worker thread pool based on the current status of worker thread pool so as to deal with more requests through using the rest of game server's resource. Fig 2 shows about internal operation of Pool Manager Thread. Pool Manager Thread can tune the size of Worker Thread Pool.

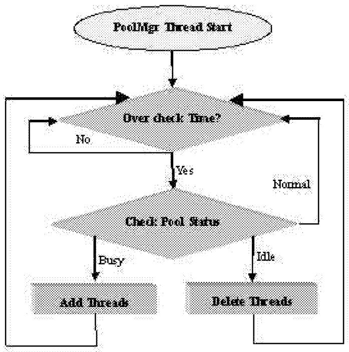


Fig. 2. Flow chart of pool manager thread operation

It is unique and works through checking the status of worker thread pool (Busy, Idle and Normal) at intervals of a check time. If the status of worker thread pool is busy, pool manager thread adds some new worker threads. If the status of worker thread pool is idle, pool manager deletes some free worker threa ds. If status is normal, there is not any change in worker thre ad pool until next check. Dynamic Thread Pool avoids the management overhead of too many thr eads because it can delete extra worker threads by itself. And it can increase the performance because it can add several worker threads when most of worker threads are busy to work and there are still many requests waiting for processing. However, because its variance of thread pool size is fixed and the amount of clients' requests changes constantly, tho ugh worker threads added into the dynamic thread pool can increase the ser ver performance of processing requests, som etimes they are not enough. To solve this problem, exponential average can be used to predict the amount of worker threads in next time. If we can predict how many extra worker thre ads is needed in next time, we can save creating time and avoid process those packets until the idle worker thread is created when there are big amount of packets arrived. And we can also reduce wasting in system resource because thread waste a part of memory and CPU scheduling time when too many idle wor ker threads still exist. Today, predictable dynamic thread pool has been advanced. Before tuning the thread pool size, game server program first predicts the amount of threads needed in the next time thr ough using exponential average idea [3] [5] in favor of increasing the server performance and saving the extra system resource waste. Exponential average approach [5] is used in the CPU scheduling problem for the prediction of the idle period. In the CPU scheduling problem, operating systems need to predict the length of the next CPU burst in order to make appropriate process scheduling. In general, the next CPU burst is predicted as an accumulative average of the measured lengths of previous CPU bursts. Similarly, we can predict the next idle period by the accumulative average of the previous idle periods. Exponential average formula is

$$\Gamma_{n+1} = \alpha t_n + (1-\alpha)\Gamma_n \quad (0 \le \alpha \le 1)$$

The parameter α controls the relative weight of recent and past history in the prediction. If $\alpha = 0$, then $\Gamma_{n+1} = C$. In other words, the recent history has no effect. On the other hand, if $\alpha = 1$, then $\Gamma_n = t_n$. In this case, the prediction only takes into account the most recent running threads but ignores the previous predictions. In our implementation, α is set to be 0.5 so that the recent history and past history are equally weighted.

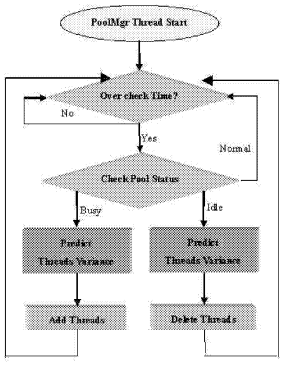


Fig. 3. Flow Chart of Pool Manager Thread operation with predicting variance of worker threads

3 Proposed system and experimental results

3.1 Proposed System

Fig 3 shows internal operation of pool manager thread in addition to the function of predicting how many worker thre ads need be added/deleted based on the amount of worker threads in the next time when the status of worker thread pool is busy or idle.

This figure is full operation of pool manager thread in our proposed system. There are two problems to realize proposed system to online game server programming. Problem 1 is whether proposed system must be better than static thread pool for any amount of users or not. Problem 2 is how to determine check time. That is to say the frequency of checking the status of worker thread pool. If check time is too small, game server program has to add/delete some threads very frequently. It brings so mu ch Creation /Destroy of thread as to spend much time on creating/destroyin g thread instead of processing game pac kets. The benefit of adding worker threa ds is canceled by the overhead of threads' creation. If check time is too big, the Pool Manager Thread can not monitor the status the dynamic thread pool effectively. To judge that dynamic thread pool fits what kind of online game server and describe the effect of check time for improving the performance of game server program, I do several groups of experimentations with different user amo unt and tune the check time value in every group. The game server operating system is Windows 2000 Server running on two 2GHz Intel(R) Xeon(TM) processors. The physical memory size is 2G. Game server's I/O model is IOCP. And clients run on one 2.8GHz Intel(R) Pentium(R) 4 processor and physical memory size is 512MB. They are 800 users and 1000 users. We run 200 threads (every thread simulates one client) on one client PC at best. A new thread will be created every one second. Every client sent 3 kinds of different packets. Sending method is as follows. The whole testing time is 3 minutes.

3.2 Experimental results for problem 1

We did 10 tests about proposed system with different check times (they are from 100ms to 1000ms) and one test about static thread pool. These following 3 figures are comparison diagram s of game server with dynamic and static thread pool as follows.

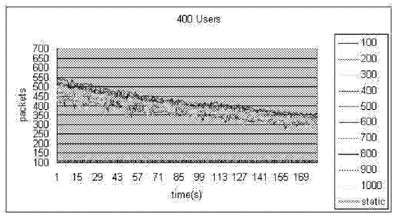


Fig. 4. Proposed/Static Thread Pool for 400 Users

Fig 4 shows the experimental result that the proposed system does not have difference with static system.

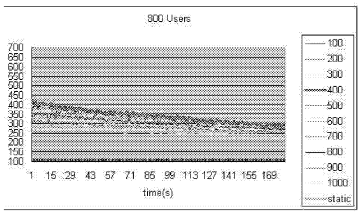


Fig. 5. Proposed/Static Thread Pool for 800 Users

Fig 5 shows that the effect of static thread pool and dynamic thread pool is different. We can see the performance of dynamic thread pool is much better than the one of static thread pool when check time equals 0.4s.

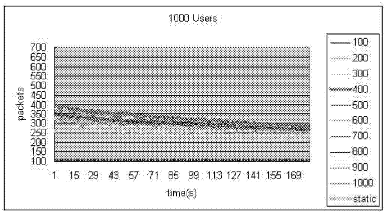


Fig. 6. Proposed/Static Thread Pool for 1000 Users

Fig 6 also shows that the effect of static thread pool and dynamic thread pool is different. This time we also can find the performance of dynamic thread pool is much better than the one of static thread pool when check time equa ls 0.4s again. Through these experimentations above, we find that the dynamic thread pool doesn't make much difference to the performance of game server using static and proposed system when the amount of users is 200, 400 and 600. However, the dynamic thread pool starts to make much difference to the performance of game server when the amount of game users is 800 and 1000. And when check time of dynamic thread pool is modified properly, the effect of dynamic thread pool is the best.

3.3 Experimental results for problem 2

Because the check time of dynamic thread pool give the effect to the performance of game server using dynamic thread pool, we listed the average packets processed by dynamic thread pool when the amount of users is 800 and 1000.

We find that the performance of game server is the best when check time equals 400ms in the Fig 7 and Fig 8 above.

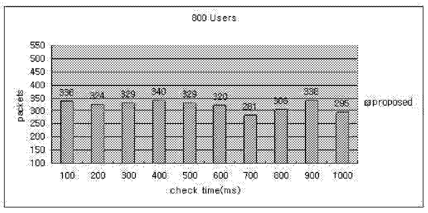


Fig. 7. Average Packets processed by proposed system for 800 Users

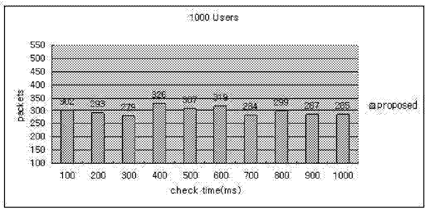


Fig. 8. Average Packets processed by proposed system for 1000 Users

4 Conclusion

In this paper, I built a dynamic thread pool system for online game server program based on exponential average. It can check the status of worker thread pool at intervals and predict how many worker threads are needed in the next time and add/delete proper worker thre ads for improving the performance of game server program and saving the game server's resource. From experimental results, the dynamic thread pool system can increase the throughput of game server only when the amount of game user is huge. If there are not so huge game users, it is possible in using static thread pool or dynamic thread pool. In reverse, it is better in using dynamic thread pool in game server program wh en the scale of online game is big. And when we use dynamic thread pool system in online game program, we need to tune the check time in order to obtain the best performance for game server program.

References

1. Cox, T., Engineer, L. , *Windows 3rd Party Gaming Group*, Multithreading for Windows Multiplayer Gaming Servers
2. Jones, A., Ohlund, J. , *Network programming for Microsoft windows, Second Edition*
3. Jung,J.J., Han, S.Y., Park, S.Y. , *Prediction-based dynamic thread pool model for efficient resource usage*, Computer Systems and Theory, Vol. **31** , No. 4 (2004) 213-223.
4. Ling, Y., Mullen, T., Lin, X. , *Analysis of Optimal Thread Pool Size*, ACM SIGOPS Operating System Review, Vol.**34**, No.2 (2000) 42-55.
5. Peterson, J. L., Silberschatz, A., *Operating System Concepts, 2nd Ed.* 118-120, Addison-Wesley Publishing Co. Inc.
6. Richter, J. , *Advanced Windows. 3rd Edition*, Microsoft Press, 1996
7. Xu, D., Bode, B. , *Performance study and dynamic optimization design for thread pool system*, preprint (2004)

Biography

Soon-Jeong Ahn was born in Korea and went to the Ewha Womens University in Seoul, where she obtained her Ph.D degrees in 2003. At the same year, she employed at Research center for advanced science and technology in Dongseo University. She is researching about the technology for 3D animation and game.

Hierarchical Solution to Scalability Issues in P2P MMOG

Abdennour El Rhalibi, Madjid Merabti
School of Computing and Mathematical Sciences
Liverpool John Moores University
Liverpool, UK
a.elrhalibi@ljmu.ac.uk ; m.merabti@ljmu.ac.uk

Abstract

MMOG are very large distributed applications, sharing very large states, and supporting communication between potentially thousands of player nodes. Despite the development of many solutions to define suitable architecture, communication protocol and enabling efficient deployment of these types of applications, many issues remains which still require a solution. In this paper we discuss MOG deployed over a Peer-to-Peer architecture, supporting a distributed model of systems with shared state and we address issues related to scalability, interest management and communication. We identify an efficient partitioning and distribution of the shared state as an important aspect in such models and propose a hierarchical multi-level interest management algorithm which enables contextual communication between peers. Experiments have been carried out and show the performance of the approach.

1. Introduction

The past few years have been very profitable to networked multiplayer games. PC, console and even mobile phone games have benefited from the increasing bandwidth and possibilities, technology and internet providers are offering. Nowadays, networked multiplayer games are going from the simple turn-based card games like Microsoft Hearts to massive multiplayer online games with hundreds of thousands of users playing at the same time like in the recent UltimaOnline [1], EverQuest II [2], and World of Warcraft [3].

Games such as flight simulators, first person shooters (FPS), massive multiplayer online games (MMOG) and real time strategy (RTS) games have high requirements in maintaining the consistency of the virtual world. However, today's Internet protocols are poorly suited for games because they need security and very fast transmissions [4][5].

The paper discusses the feasibility of using peer-to-peer (P2P) overlays to support a typical persistent world MMOG, as a replacement for the client/server model. The technical details of these topologies will be discussed, along with the issues. We will discuss the possibility to exploit the hierarchical and distributed nature of P2P architecture to provide an efficient scalability scheme and a suitable P2P interest management algorithm [9] which could reduce greatly the amount of data exchanged in the network.

The rest of the paper is organized as follows: section 2 provides background information about current MMOG issues, section 3 presents the Peer-to-Peer architecture we are experimenting to deploy in a MMOG P2P overlay, section 4 introduces the details of the interest management we propose, and discusses experiments to evaluate scalability, and volume of data exchanged Finally in section 5 we conclude and expose the future work.

2. Issues in MMOG Development

2.1. Scalability Issues for MMOG

The appearance of networking environments provides a platform for the development of gaming systems which can be accessed by many players simultaneously. The Internet is the main platform for the generalized adoption of this kind of systems, making it possible for a large number of users to simultaneously and cost-effectively access these systems, and for designers to consider the development of massively multiplayer gaming experiences, and thus raising the question of scalability.

We define scalability as the ability a system has to deal with increasing demand from the users, without significantly degrading the level of interactive experience. So, quality of service decay in a controlled way, the removal of bottlenecks, the extension of system resources to deal with increasing demand and technology updating to avoid becoming outdated, are all challenges in the design of a scalable solution [4][6][7].

Considering the characteristics of MMOG, we will define the main scalability issues that these systems must consider along the following considerations:

• game state simulation component that allows thousands of players to share the same virtual event or interactive context;

• storage capacity for all the data that is used to represent the virtual environment (data model and multimedia elements) and its efficient and timely distribution;

• communication performance that enables the interactive system to maintain the quality and coordination of the game experience;

• an overall architecture that enables the integration of new components, resources and technologies for system extension.

Next, we will present a discussion of some these issues and the techniques relevant to deal with these requirements.

2.2. MMOG Architecture Issues

The issues related to traditional architecture used for MMOG are well known [4][5], and we are going to discuss some possible solutions.

Although the majority of existing networked systems are based on the Client/Server architecture [8][9], there is another architecture worth of consideration, the Peer-to-Peer architecture.

Peer-to-Peer architectures are gaining ground over client/server, as more experiences are made and some of its problems are overcome. Recently various distributed systems emerged based on this architecture, with file sharing as the most common application [10]. The idea for Peer-to-Peer architecture is based on the concept of a system where all components are treated as equally important and thus avoiding the scenario where a specific resource becomes more critical and the overall system performance dependent on it [11]. Another characteristic of Peer-to-Peer architecture is the design that tries to take advantage of the existing resources available on the Peers' – load or processing capacity, bandwidth capacity and storage capacity – and use them for common benefit.

One of the possible solutions to achieve scalability using P2P architecture, is based on game space subdivision. Each of the resulting game regions is attributed to a peer that became responsible for simulating or communicating the events that occur in that region [12] thus taking advantage of each peers' load capacity. In this approach some of the challenges known for client/server architecture also applies and others appear. Firstly, in the process of attributing a game region to a peer, it is important to take into account some measure of load capacity, memory and network bandwidth, to avoid to allocate a game region management to a peer that does not have the necessary resources thus damaging the quality of gameplay for all players in the sector. Other challenges are inherent to Peer-to-Peer architecture and relate to the unpredictability that is typical of the P2P networks, with the arbitrary joining and leaving of peers. While we can manage to have backup servers to compensate for those leaving, this feature introduces a new problem in keeping simulation object addressing updated and simulation data persistent. There is already some work being done in this area [13][14].

Another solution that is based on the P2P architecture and is more drastic than the previous one, is to have no central simulation. In this case we would need a distributed simulation model where all peers would have to virtually simulate the space where the player is interacting [15][16]. This is the approach we advocate in our paper.

3. Peer-to-Peer Architecture Proposed

3.1. A Peer-to-Peer Architecture for MMOGS

Considering the scalability requirements discussed in section 2 we propose an approach to the challenge of balancing these for the purpose of designing a generic MMOG architecture [15]. For this approach we will need to analyze the dependencies between the requirements, we have considered as the basis for this approach a Peer-to-Peer architecture with a distributed simulation model where each peer computes, locally, the game region where the player interacts. A distributed peer-to-peer simulation brings clear advantages by lowering the cost of centralized infrastructures and by distributing the processing load to where it is benefited from, while complexity increases for maintaining the consistency and persistence of virtual world [15].

3.1.1. A P2P Topology. The topology we propose is a hybrid solution starting with an initial architecture based on a main server, and building-up a P2P topology as the number of players increases (Figure 1). As the players connect to the game, the server delegates more and more of its role as game and network/communication manager to the connected player machines, which self-organize in a P2P fashion. The peers still rely on the initial server to join and leave the game, and to help them discover their peer-group if already created and to receive the game data when a new region is required. However all the in-game communications once the players is connected to his peer-group are done in P2P fashion [15].

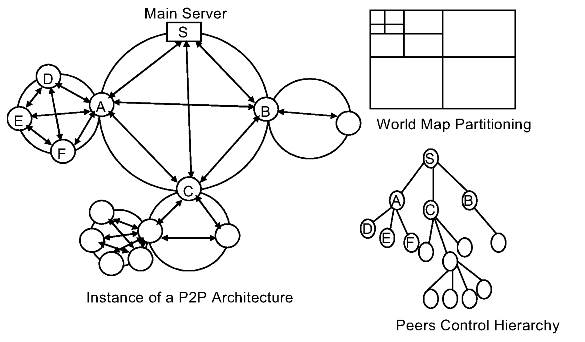


Figure 1. Proposed Architecture

The architecture allows the developers to maintain direct control and authority over the players account information. Once a joining player has been checked, he will granted connection to his peer-group and will be managed in a P2P game session until he leaves the game.

The architecture is flexible, robust and dynamic. Several spatial data structures such as BSP, octrees, or PVS can be used to control peers group and their relations in a dynamic way, and to map the peer-groups.

In the following sections we discuss the meta-model architecture and the P2P protocol for joining and leaving the game. This protocol is the most important as regards to the P2P architecture.

3.2. Meta-Model Architecture

The first step in our P2P architecture and protocol development is the design of an agent based meta-model architecture. We have defined a high level design based on mobile agent model and suitable for the development of the MMOG and a simulation environment. Our meta-model architecture provides the rules to develop a simulation environment for MMOGs, and to implement an instance of P2P protocol for MMOGs. Using a mobile agent model we can represent all the dynamic and static aspects of an MMOG [15].

To support multi-agent organization, communication and coordination as a P2P infrastructure, we also incorporate the concept of agent groups. Any agent within the agent group may perform multicast or subcast. A multicast communication allows an agent in the group to send a message to all other agents in the group, no matter where the agents are in the system. A subcast enables an agent to send a message to a subset of the group.

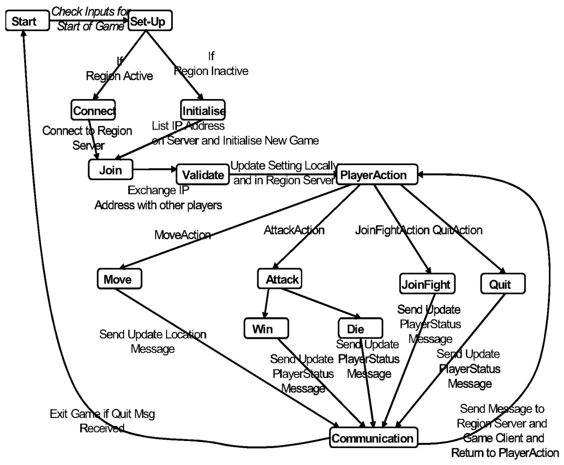

Start
Check Inputs for Start of Game
Set-Up
If Region Active
If Region Inactive
Connect
Initialise
Connect to Region Server
List IP Address on Server and Initialise New Game
Join
Validate
Update Setting Locally and in Region Server
PlayerAction
Exchange IP Address with other players
MoveAction
AttackAction
JoinFightAction
QuitAction
Move
Attack
JoinFight
Quit
Win
Die
Send Update Location Message
Send Update PlayerStatus Message
Send Update PlayerStatus Message
Send Update PlayerStatus Message
Send Update PlayerStatus Message
Exit Game if Quit Msg Received
Communication
Send Message to Region Server and Game Client and Return to PlayerAction

Figure 2. Agent communication level – an example

3.2.1. P2P Protocol for Joining/Leaving. Below we present briefly the main scenarios in the P2P protocol for joining and leaving the game [15]:

- In the initial state there are no players, no peers, only the main game server (world server) maintaining the full game, the players database, the current game data for existing players, etc…
- If a new player joins the game, he will be connected to the world server and assigned the role of new region server for the region he will play on. He will be sent all the data required to maintain the region, and the identity of any other currently running region server. Any new player joining in the same region will be connected in P2P fashion to the peer-group managed by the region server. The region server will be sent the data of the game, the data of the players joining, and the data of the others existing regions server controlling the others regions.
- If a player moves to another region we have two situations which might occur. Firstly if the region is already controlled by another server region, the new player will join the peer-group associated to this server-region. Secondly if no peer-group for the region exists, the client machine will be assigned the role of server region.
- If a player leaves the game. We have again two cases. If the player is a client in his region peer-group, it will be simply disconnected and the peer-group will be informed. If the leaving player is a region server, the protocol will elect a current peer

(client) to become the new region server. The connection with the world server will be re-established. If there are others regions server they will be informed of the disconnection of the leaving region server, and will be given the identity of the new region server. Only the region servers are connected to the world region (i.e. the main server).
Figure 2 shows an example for the agent communication level, part associated to a peer. Note in particular the states Join and Quit which implement the protocol described in this section. The interest management stage is discussed in section 4.

4. Interest Management and Experiments

Interest management [18] is an important technique which enables scalability and reduces communication. As the scale of an MMOG increases in terms of number of players (clients or peers), and game objects, world state and information broadcasting will consume important network resources. Therefore techniques such as interest management, which work as message and data filtering mechanism, become very important to guarantee scalability. The basic idea of interest management is relatively straightforward: All clients or peers should receive only the information from the game that concerns them, rather than all the information related to the whole game world state and changes. Changes to shared environment or game objects are automatically made available only to involved clients / peers.
In the P2P architecture we have proposed there is a natural structure which can be used to reduce the information to communicate between peers. The hierarchical partitioning of the game world in region, controlled by a region super-node (which is a region server, but also a peer), and forming a P2P network for this region guarantee that only the information updates occurring in this region will be sent to /exchanged by the peers mapped in this region. As noted in above, many spatial partitioning techniques can be used, based on Octree, BSP or PVS, and all require the dynamic management of a subscription list [12][17], to enable to peers to join a region network, identify the region super-node, and identify the peers' addresses.
However, as a node joins a region, the size or resolution of the region chosen will not guarantee that all the information communicated to a peer are all relevant. A further stage of data/events filtering is required to reduce further the messages exchanged. The mechanism used is based on aura detection [17], and on line of sight/perception, or contextual line of sight (i.e. considering different level of awareness).
When a peer joins a region, it is elected as super-node to manage the interest management and data filtering mechanism for the region, or simply joins as peer. Upon joining it subscribes to region subscription list and gets a copy informing it of the peer network members. It sends an update of its state to the super-node which determines the intersection of all the peers' aura and contextual line of sight, calculating cliques of peers sharing the same interest and line of perception, and updates the subscription lists to be sent to all the peers' clique members.
A new level of partitioning is applied to create a sub-region with a new elected sub-region super-node which will create a new subscription list and manage the sub-region P2P network in which multicast communication is used between peers of the same clique, to exchange in-game messages.

We exploit the recursive properties of the spatial partitioning of the game world to create different level in a hierarchy of communication, where full multicast is only used at the lowest level of the hierarchy. This involves only peers belonging to the clique determined by peers' aura intersection and line of perception determination. The mechanism makes the interest management and data filtering requirement distributed. It is managed by different super-nodes at different level of the hierarchy, and scalable with the management of hierarchical list of subscription which supports a full multicast only at the lowest level.
We have carried out 100's of experiments to forecast and compare the performance of client/server with interest management and our peer-to-peer architecture with/without interest management controlled by regions and data-filtering controlled by contextual line of sight. The spatial partitioning used is based on BSP with a 3 level hierarchy, and calculation and update of clique is done upon joining and leaving a region.

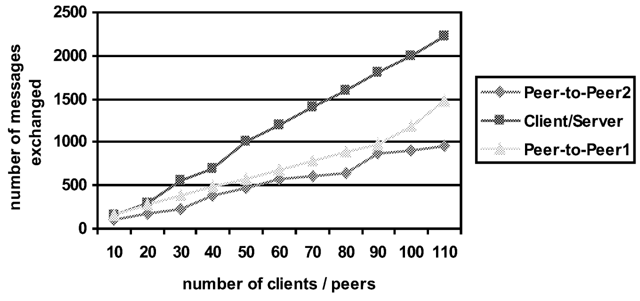


Figure 3: Performance Test Simulations

We have compared client/server vs. P2P with (Peer-to-Peer2) and without (Peer-to-Peer1) interest management. The results show that interest management, combined with data filtering mechanism provides better performance to ensure scalability and reduce the number of the messages exchanged.

5. Conclusions

Despite the development of solutions to define suitable architecture, protocol and enabling efficient deployment of MMOG applications, many issues remains which still require a solution. In this paper we discuss MMOG deployed over a Peer-to-Peer architecture, supporting a distributed model of systems with shared states and we address issues related to scalability, interest management and communication.

These issues and the integrated approach which has been presented are being tested and developed with the following questions - what should be the…?:

- Best spatial subdivision to use, which will depend on the nature and complexity of the game world. Outdoor vs. Indoor, large room vs. corridor, perception limited to visual vs. extra-sensorial perception. The spatial partitioning techniques choice is important as they provide different performance and accuracy.
- Region resolution and size. It is related to spatial partitioning, but also determines the size of the networks, and has a great effect on the number of communication required, and load balancing.
- Number of level in the hierarchy. It is related to the region size, and determine the required overhead to calculate regions, sub-regions,…, and cliques.
- Size of subscription list. It is determined by all the above and contribute to scalability and overall communication performance.
- Cycle of clique determination and update. It can be done upon joining and leaving a clique/region, or at regular time interval. It contributes to the overall performance and scalability of the dynamic and distributed deployment of the peer-to-peer network.

The research is at its first stages, and many problems are still open and will constitute our future work.

6. References

[1] Electronic Arts, UltimaOnline; http://www.uo.com

[2] Sony Online Entertainment INC., The EverQuest II homepage; http://everquest2.station.sony.com/

[3] Blizzard Entertainment, The World of Warcraft homepage; http://www.worldofwarcraft.com/

[4] Coulouris, G., J. Dollimore, and T. Kindberg, Distributed System: Concepts and Design, Addison-Wesley, England, 2001.

[5] J. Smed, T. Kaukorante and H. Hakonen, “Aspects of Networking in Multiplayer Computer Games”, International Conference on Development of Computer Games in the 21st Century, Hong Kong SAR, China, November 2001.

[6] P. Jogalekar, “Evaluating the Scalability of Distributed Systems”, IEEE Transactions on Parallel and Distributed Systems, 2002 IEEE, Vol. 11, No. 6, June 2000, pp. 589-603.

[7] A.B. Bomdi, “Characteristics of Scalability and Their Impact on Performance”, Proceedings of the second international workshop on Software and performance, ACM Press, Ottawa Canada, 2000, pp. 195 - 203.

[8] P. Rosedale and C. Ondrejka, “Enabling Player-Created Online Worlds with Grid Computing and Streaming”, September 2003, Available at: http://www.gamasutra.com/resource_guide/20030916/rosedale_01.shtml

[9] L. Aarhus, K. Holmqvist and M. Kirkengen, “Generalized Two-Tier Relevance Filtering of Computer Game Update Events”, Proceedings of the first workshop on Network and system support for games, ACM Press, April 2002, pp. 10-13.

[10] Gnutella.com Inc., Gnutella; http://www.gnutella.com/

[11] Oram, A., Peer-to-peer : harnessing the benefits of a disruptive technology, O'Reilly, 2001.

[12] T. Limura, H. Hazeyama and Y. Kadobayshi, “Zoned Federation of Games Servers: a Peer-to-Peer Approch to Scalable Multi-player Online Games” SIGCOMM’04, ACM Press, August 2004.

[13] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, “Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing”, Internal Report, Computer Science Division, University of California, April 2001.

[14] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, and B. Zhao “OceanStore: An Architecture for Global-Scale Persistent Storage”, ACM, 2000.

[15] Abdennour El Rhalibi and Madjid Merabti, “Agents Based Modeling for a Peer-to-Peer MMOG Architecture”. ACM Computer in Entertainment Journal. April 2005, Edited by Newton Lee – Editor-in-Chief of ACM CiE.

[16] C. Diot and L. Gautier, “A Distributed Architecture for Multiplayer Interactive Applications on the Internet”, IEEE Networks magazine, IEEE, July/August 1999, pp. 6-15.

[17] H. Abrams, K. Watsen, and M. Zyda, “Three-tiered interest management for large-scale virtual environments”, Proceedings of 1998 ACM Symposium on Virtual Reality Software and Technology (VRST’98), ACM Press, November 1998.

[18] K. L. Morse, “Interest management in large-scale distributed Applications” report, Department of Information & Computer Science, University of California, Irvine, 1996.

LATE PAPER

Harnessing Agent-based Games Research for Analysis of Collective Agent Behaviour in Critical Settings

Abdennour El Rhalibi, A. Taleb-Bendiab

School of Computing and Mathematical Sciences
Liverpool John Moores University

a.elrhalibi@livjm.ac.uk ; cmsatale@livjm.ac.uk

ABSTRACT

Despite the complexity of modern computer games and the wide range of different themes explored, there has been only a few games that feature large chaotic groups of people, such as those found at riots or protests. The most popular, *State of Emergency*, features hundreds of individual people on the screen at any one time, however the Artificial Intelligence (AI) that controls the rioting NPCs is fairly simplistic – the civilians run in seemingly random directions, unaffected by one another and don't seem to have specific objectives. In real protest and riot situations, each individual person involved has many factors that dictate their behaviour. Mood, temperament, behaviour of surrounding people and any perceived danger are all among the important aspects of the situation.

This paper gives a brief overview of an AI mechanism that has been developed specifically for controlling riots and protests in games. The model is based on previous research into Emotional Societies and presents a realistic and believable environment for games, which can operate effectively with a relatively minimal impact on resources.

INTRODUCTION

In recent years, there has been a marked increase in the number of games that feature large groups of artificially intelligent entities. However, these tend to be military strategy games such as *Cossacks* or the *Total War* series, where groups of hundreds of soldiers make up squads and divisions. Because these kinds of large groups of troops are well disciplined in real-life, their actions are predictable and the AI that controls their behaviour in the games can be simpler and still maintain a high level of realism. Large groups of people that don't follow military discipline, such as crowds of individuals in a public place, are a lot more chaotic. In real crowd situations, each individual person involved has many factors that dictate their behaviour. This includes mood, temperament, behaviour of surrounding people, perceived danger, and a number of other aspects of the situation. Due to these reasons, there have only been a handful of games that have attempted to tackle that environment, and have achieved varying degrees of success.

A significant amount of research has already been done into developing better AI for managing crowds. However, this majority of the research in the field is usually aimed at recreating purely chaotic and random crowds. There is very little done to develop real-time AI simulations of disorganised crowds with common objectives, such as is found in riot or protest situations. These crowds share a number of common, unspoken objectives. In the case, for example, of a peaceful student protest against tuition fees, those objectives are likely to be:

1. Gain as much attention as possible
2. Maintain the protest for as long as possible

Individuals within the crowd may have their own objectives in addition to these, and may even work against those goals, such as vandals wishing to cause damage to property – this activity increases the likelihood of police resistance force and indirectly works against the second goal as police stop the protest early. Although riots and protests are certainly not the only types of crowd that have goals or common objectives amongst their participants, they are far more interesting in a computer game environment than, say, a tour group in a museum.

THE MODEL

The primary objective of research into riots and protests was to identify a simple model that describes how human beings operate in a riot or protest situation, and that answers the most pertinent questions- What stimuli are there in a riot or protest? How does a human respond to this? What kind of interactions are there between people in riots or protests? How does this affect their behaviour?

The model that has been developed is described briefly in this paper and at more length in [6]. The mechanism for controlling the riots is based on research into a number of fields, including AI (Rule Based Systems and Fuzzy Logic especially), Multi-Agent Systems (MAS), Chaos Theory and Cellular Automata. The rules that drive the model are based on research into riots and protests, including police public order training manuals [4], interviews with officers from the Merseyside Police and personal accounts of riot and protest experiences.

EMOTIONAL MODEL

Each artificial agent in the riot control mechanism is treated as an individual humanlike entity. That is, decisions in movement and behaviour for that agent are based entirely on a humanlike reasoning processes for that agent alone, rather than a pattern-based system that dictates movement and behaviour for many agents at once. The mental model for the agents in the system is based loosely on research into emotional societies by David Chaplin [2], who defined a model for emulating a human society that involves such concepts as mood, social responsibility and inter-agent relationships.

Each agent has a different "mood". This mood is a collection of values that represent different emotions the progression of the simulation. These emotions represent the way the agent *feels* at any given point during the simulation. Like the fixed attributes, they are set initially based either on random values, or patterns observed in real life.

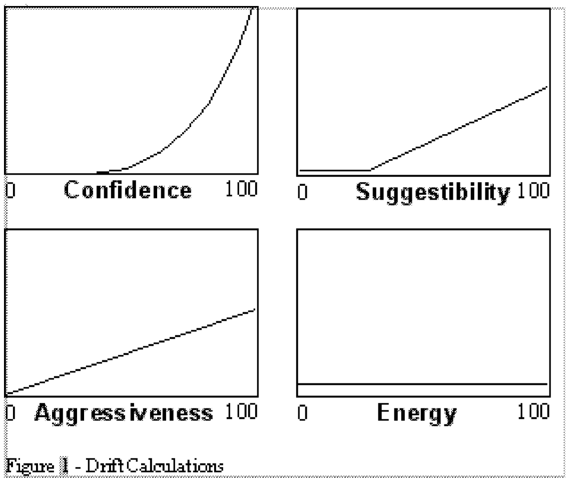


Figure 1 - Drift Calculations

- **Confidence** – *How confident and secure they feel in the current situation*
- **Suggestibility** – *How easily they are affected by the behaviours of others*
- **Aggressiveness** – *How aggressive the agent is feeling at the current time*
- **Energy** – *How much energy the agent has left, how long will they will stay involved*

All emotions are variable, but only to a certain extent. "Drift" is an additional mechanism that is included to stop emotions from being changed too far from their original values (i.e. An individual behaving too far out of character). Every time the agent is updated, all emotions tend back towards their original state, at different rates based on the difference between the original and current values (Figure 1).

AGENT ARCHITECTURE

Figure 2 illustrates the agent architecture behind the riot and protest control mechanism. It is fairly self explanatory. There are two main process that occur when updating the agent's state:

Evaluation of the environment, in which the agent evaluates the behaviours of other agents in the crowd who are within two certain ranges, the attributes local environment and their current emotional state. The result of this evaluation is to see if these factors affect the emotional state, and therefore perhaps the behaviour, of that agent.

Reaction to the mood, where the agent chooses an appropriate behaviour to exhibit based on their current mood. The choice of behaviour affects their movement, and in turn the evaluation process of other agents, thereby emulating the non-verbal emotional communication seen in riot or protest situations.

Mechanically, both processes are resolved using a combination of Fuzzy Logic and Rule Based Systems. Emotion and attribute values of the mood of the agent are sorted into different fuzzy sets, depending on their current state. In the evaluation phase, the different sets are combined with facts about the environment taken from the sensors to create a current "knowledge base" (KB) for that agent. This KB is fed into a RBS (CLIPS is used in the example implementation) which contains a list of fixed rules. These rules describe the emotional

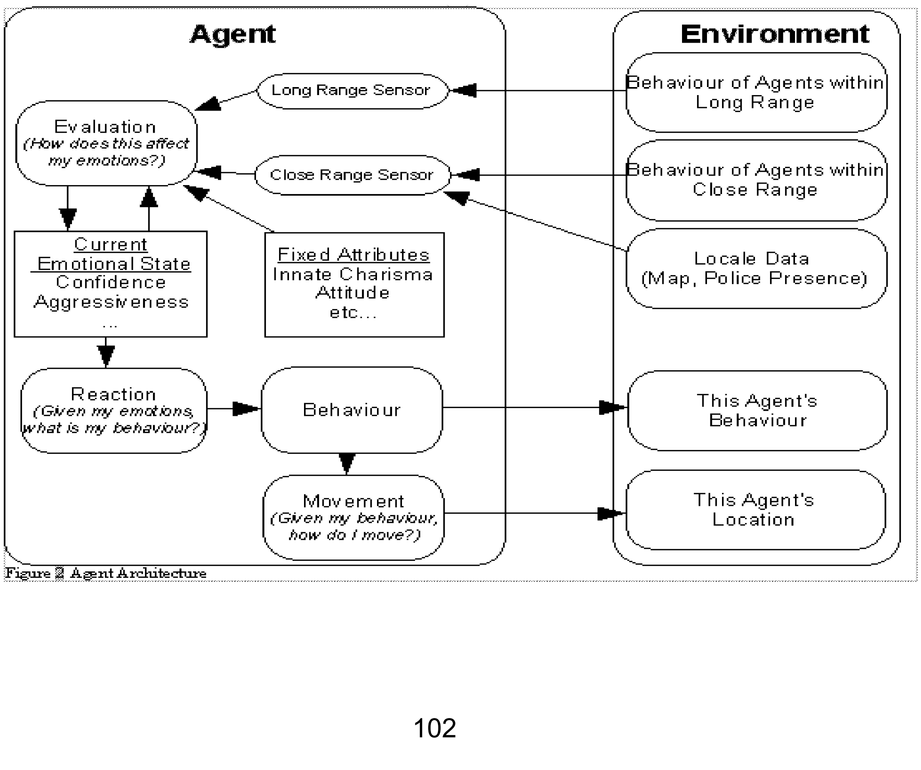

Agent
Environment
Long Range Sensor
Evaluation (How does this affect my emotions?)
Close Range Sensor
Behaviour of Agents within Long Range
Behaviour of Agents within Close Range
Current Emotional State Confidence Aggressiveness ...
Fixed Attributes Innate Charisma Attitude etc...
Locale Data (Map, Police Presence)
Reaction (Given my emotions, what is my behaviour?)
Behaviour
This Agent's Behaviour
Movement (Given my behaviour, how do I move?)
This Agent's Location

Figure 2 Agent Architecture

changes an individual goes under given specific circumstances. For example:

TABLE 1 EXAMPLE RULES

Rule	*Condition*	*Action*	*Salience*
Flee	Suggestibility>=HIGH, CloseAgent=FLEE_IN_TERROR	Confidence-=10, Suggestibility+=5	0.8
Seen Arrest	Confidence <HIGH, CloseAgent=ARRESTED	Confidence-=5, Suggestibility -=5	0.4
Incited to Violence	Aggressiveness >=LOW, Suggestibility=HIGH, Confidence >MINIMUM, CloseAgent=ATTACK_INCITE	Confidence +=3 Aggressiveness +=15	0.3

In the reaction process, the rules are simpler – for each possible behaviour in the riot or protest, there is a specific set of mood values that trigger that behaviour. Behaviours are mutually exclusive, so that only (and always) one behaviour occurs as the result of any given combination of mood values. The exact rules and fuzzy logic definitions can be found in [6].

MOVEMENT

The movement of agents in the riot or protest simulation is possibly the most important factor for better games – the movement is the most visible external effect of the model. The movement is performed using a simple algorithm based on a technique called Orchid Fractal Analysis [8] (OFA). OFA was developed at Warwick university to model crowd movements into and out of large stadia. There are obvious similarities between these situations and riots and protests, and OFA proved a very useful base to build the movement algorithm on. The algorithm itself is expressed as a flow chart in Figure 3.

The algorithm includes **flocking** based on agent confidence, as agents with low confidence follow other agents with similar behaviours. OFA has been modified to allow **patience**, as agents will not wait too long in a static crowd if they wish to move past it. Movement destinations are chosen based on perceived risk of a given location.

RISK ASSESSMENT

For an individual moving around a riot or protest situation, precise destinations of movement are not important if the person is trying to stay with the crowd. They will willingly allow themselves to move with the flow of a moving crowd while they still want to be an active participant. Despite this, crowds in riots or protests are self-ordering to an extent. It is clearly observed in [5] and [7] that more passionate and excited individuals tend towards the leading edge of a crowd, and those with wavering support tend to be further back.

Therefore, the model is able to automatically rate locations based on assessed risk. This allows individual agents to select preferred locations based on their current mood – confident and passionate individuals tend to move towards the leading edge (a high risk area) while nervous individuals move away from the action.

Each location in a "world" using the model is rated for risk based on a number of factors – proximity to police units, behaviour of nearby agents and the presence of any tear gas or other environmental variables. Four sample ratings are described in Table 2.

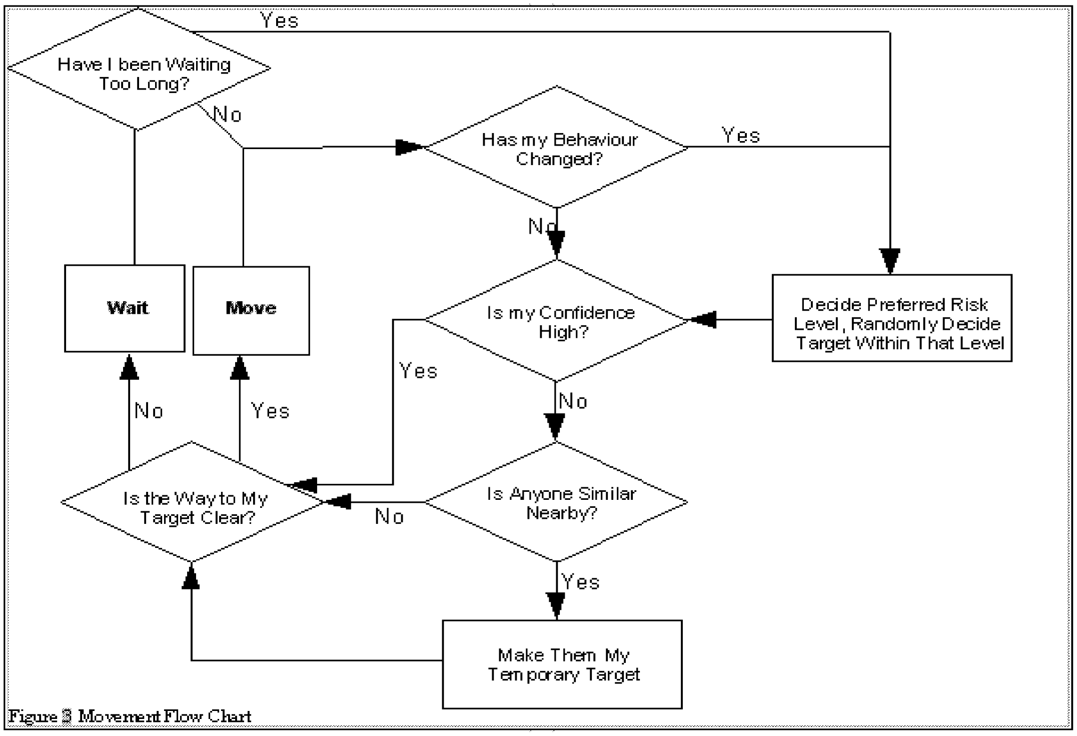
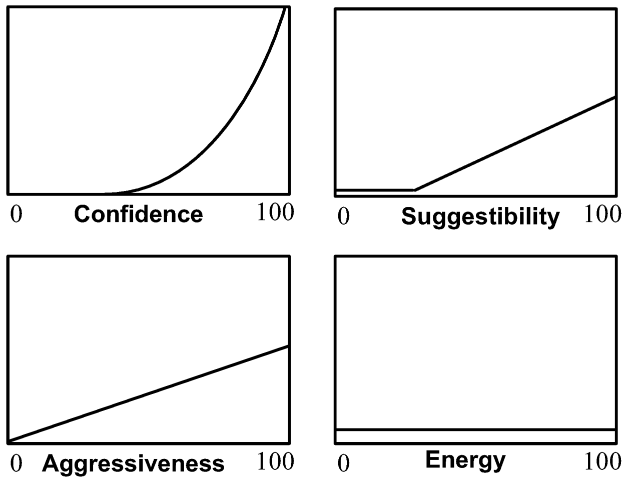

Have I been Waiting Too Long?
Yes
No
Has my Behaviour Changed?
Yes
No
Wait
Move
Is my Confidence High?
Decide Preferred Risk Level, Randomly Decide Target Within That Level
Yes
No
No
Yes
Is the Way to My Target Clear?
No
Is Anyone Similar Nearby?
Yes
Make Them My Temporary Target

Figure 3 Movement Flow Chart

TABLE 2 ZONED BEHAVIOURS

Risk Level	***Behaviours Present***
BLUE	Innocent Bystanders and other Non-participants
GREEN	Participants that are an active part of the crowd, but are "keeping out of trouble"
YELLOW	More enthusiastic individuals, leaders and instigators that form the core believers in the crowd
RED	More aggressive and forward individuals that may be violent or resistant to force.

Figure 4 shows a simple illustration of how the world may be rated. In this example, agent X has a current mood which dictates that he would prefer to get closer to the leading edge of the riot, so the algorithm chooses a target location that has the appropriate risk level associated with it. The precise physical location is not as important to riot participants as the risk associated with



that location.

Throughout runtime of the simulation, each agent chooses an appropriate preferred risk level based on their current emotional state and behaviour. This choice is not permanent and is re-evaluated every time the agent's mood is changed. If the agent does not already have a specific target destination within that risk level, one is chosen from random from the set of locations with the appropriate risk. When the agent reaches that target, or its behaviour changes its preferred risk, or the target location's risk level becomes altered, a new target is chosen at random as before.

CONCLUSION

This paper has given a brief overview of all the major features of the riot control mechanism. When the features are combined into a fully operating application, as has been done in the proof of concept [6] (the thesis also has much more technical details regarding this implementation), the model works very well. The system gives a good emulation of what can happen in real riot or protest situations, at least in terms of movement and behaviour. This paper has given an indication of the possibilities of this model, and the greater opportunities for exciting gameplay that riots or protests present.

Figure 5 depicts some screenshots of the 3D application supporting the agent architecture with a variant of the model involving a leader. The screenshots shows the variety of situation the agents might be dealing with, such as interacting with a different class of agents controlled by different rules, or coordinating their action to have a more efficient effect on the environment or on their common goals.

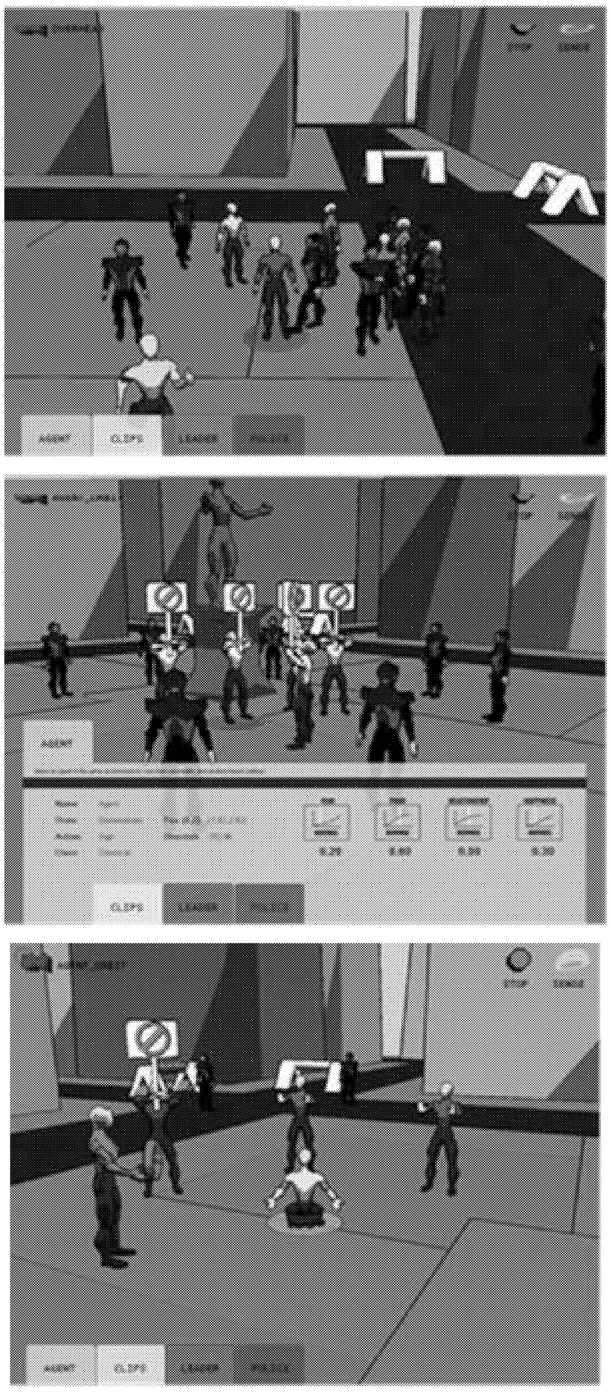


Figure 4: 3D Application supporting Protest Mechanism with Leader

Beside the potential application of the model and architecture to a computer entertainment environment, the model is generic and can be used as well for "serious" applications which involve distributed emerging behaviour, scenarios based simulation, complex agent-based modelling including emotional, reactive and deliberative reasoning. For examp le this model can easily be extended to support different emotional models, agent architectures, reasoning techniques, and application to a swarm of robots wandering in a hazardous environment.

REFERENCES

[1] Axelrod, R., (1984), *The Evolution of Cooperation*, Penguin

[2] Chaplin, D.,(2003) *Emotional NPC Societies in Games*, Liverpool John Moores University

[3] Col. Dewar, M., (1992), *War in the Streets*, BCA

[4] English, J. and English, B., *Police Training Manual (Tenth Edition),* 2003

[5] Joshua, H., & Wallace, T., (1983), *"To Ride the Storm",* Heinemann Educational

[6] Kirman, B. (2004), *Better Riots and Protests in Games: Modelling Large, Disorganised Groups of Emotionally Driven Artificial Agents,* Liverpool John Moores University

[7] Northam, G., (1988), *Shooting in the Dark: Riot Police in Britain*, Faber and Faber

[8] Still, G. K., (2000), *Crowd Dynamics*, PhD Thesis, Mathematics department, Warwick University

[9] Waddington, P.A.J., (1991), *The Strong Arm of the Law*, Oxford University Press

AUTHOR LISTING

AUTHOR LISTING